AD-A115 502 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 20/6
PROGRAMMED CONTROL OF A SINE-WAVE GRATING GENERATOR FOR VISUAL --ETC(U)
DEC 81 A L LAWSON
UNCLASSIFIED AFIT/GE/EE/81D-33 NL

1 OF 2
AD
A115502

DDC

0

AD A115502

70

UNITED STATES AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DTIC
ELECTE
JUN 14 1982
S
E

DTIC FILE COPY

This document has been approved for public release and sale; its distribution is unlimited.

82 06 14 194

①

PROGRAMMED CONTROL OF A SINE-WAVE GRATING GENERATOR FOR VISUAL RESEARCH

THESIS

AFIT/GE/EE/81D-33 Albert L. Lawson

Captain USAF

DTIC
ELECTE
JUN 14 1982
S D
E

AFIT/GE/EE/81D-33

PROGRAMMED CONTROL OF A SINE-WAVE GRATING GENERATOR FOR VISUAL RESEARCH

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Accession For
NTIS GRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes

Dist	Avail and/or Special
A	

DTIC
COPY
INSPECTED
2

by

Albert L. Lawson
Capt USAF

Approved for public release, distribution unlimited

Preface

The Aerospace Medical Research Laboratory (AMRL) is currently involved in research to understand the operation of the human visual system. Their research is dependent on the reliability of controlled tests with the test parameters tightly controlled. This thesis describes an effort to develop a device to automatically control brightness and contrast levels on a piece of visual test equipment.

Thanks to AMRL and to Major Arthur Ginsberg for sponsoring this project and for their financial support. I would also like to acknowledge the help that I received from several individuals at AFIT during my attempt to complete this thesis. Capt Lee Baker, and Capt Larry Kizer assisted a great deal in writing and debugging the Z80 assembly language program. Mr Orville Wright and Mr Dan Zambon were helpful when I had problems with circuits. I would never have accomplished much of the work without their help.

Thanks also go to my thesis committee members, Major Charles Lillie and Capt Richard Conn, for their assistance in proofreading this thesis and any assistance during the project development.

Finally I want to thank my Thesis Advisor, Professor Matthew Kabrisky for his help in both the project design, trouble shooting problems, and his assistance in proofreading.

Albert L. Lawson

Contents

Contents (cont)

Contents (cont)

List of Figures

Abstract

This describes the development of an automatic controller for an Optronix Optical Signal Generator. The signal generator is used by the Aerospace Medical Laboratory/HEA at Wright-Patterson AFB to test human visual response to sine-wave gratings. The computer used in development of the controller was a Z80 based Mostek MDX CPU-2. Sensing of the optical signal to be controlled was accomplished by a Reticon RC-100 Series Circuit Board with a G Series Array Board containing 1024 diode elements. The Reticon Array Board produces an analog video signal for each diode in the array. The computer then selects every fifteenth diode value and stores it, calculates the actual screen brightness and contrast from the stored values, and outputs a correction signal to the generator. The circuits developed include the flags required to synchronize the computer with the Reticon Card, the circuits required to input the desired values for contrast and brightness, and the circuit used to select every fifteenth diode of the array scan to be read. This controller was never fully operational, but all of the major circuits have been tested and do function.

PROGRAMMED CONTROL OF A SINE-WAVE GRATING GENERATOR FOR VISUAL RESEARCH

I Introduction

Background

Current testing of the human visual system is based on the principle of measuring visual acuity in terms of the smallest letter that can be seen. This method of testing is commonly referred to as the Snellen Method and, although effective for prescribed corrective lenses when needed, does not give an accurate measure of the visual response over the total range of spatial frequencies that the eye-brain system can process. Spectral analysis of the smallest letters on an eyechart show that the major components of small letters are high frequencies (4). Since the Snellen Method measures mainly the ability to distinguish small letters, it fails to measure visual response to low spatial frequencies. A better measure of an individual's visual response would include spatial frequency measurements which cover a broader range of frequencies. This can be done by testing an individual's visual response to sinusoidal gratings of various frequencies. Tests over these broader ranges of frequencies have produced results indicating the new test may be a better method of visual testing (2,3,6,10,and 12).

The term spatıal frequency, measured in cycles per degree, can best be expressed as the number of times a sinusoidal signal completes a cycle in one degree of visual angle. The angle is measured as a fraction of the 360 degree circle around the observer. The actual angle is determined by finding the inverse tangent of the length of the sinusoidal cycle, divided by the distance of the signal from the observer. A sinusoidal grating consists of a series of bars, usually vertical, across the field of vision. The bars can be generated conviently by sinusoidally varying the video voltage to a cathode ray tube, thus producing a series of bright columns with brightness between the columns varying sinusiodally. Contrast is a function of the difference between the bright and dark columns and the average luminance of the screen. Most testing with sinusiodal gratings is done by lowering the contrast until the screen appears blank, after which the contrast is increased until the observer can recognize a bar pattern (2,6,12). The frequency of the signal is measured as described earlier, with a cycle represented as the distance from one bright bar to the next. The point of signal recognition is called contrast threshold.

The contrast sensitivity function is a measure of an individual's contrast threshold sensitivity. The Aerospace Medical Research Laboratory (AMRL/HEA), Wright-Patterson AFB is currently studying methods of testing the human visual system using sinusoidal gratings. AMRL/HEA uses an Optronix Optical Signal Generator to provide sine-wave grating signals for visual system testing. An operator adjusts the brightness and contrast of the signal before testing, with no further adjustments made by the operator during the test session. The only check on actual brightness and contrast generated by the Optronix Generator is when the generator is optically calibrated with an external standard. Concern has been raised that luminance levels of the test signals may vary during a test or periods between calibration creating an undesired error in test measurements.

AMRL/HEA has requested that a system be developed to monitor the luminance level of the generator output and make automatic adjustments required to maintain the desired luminance and contrast level calibration. Study was started (Capt. Ken Martindale, AFIT Thesis, 1980)(8) on a method of controlling the generator output with a Z-80 Microprocessor, a Reticon Photo-diode Array, and Datel Analog-to-Digital (A/D) and Digital-to-Analog (D/A) Converters to construct a controller to monitor and control the luminance signals.

AMRL/HEA purchased equipment to build the proposed controller, however, the equipment arrived too late for Capt. Martindale to begin any circuit design or develop any computer algorithms. This thesis will cover the attempt to use the equipment already purchased to develop the desired microprocessor controller. The specific purpose is to develop a method of controlling the contrast of the signal generated by the Optronix Signal Generator. This will reduce the posibility of introducing errors in the visual system measurements.

As a first cut of the project, the automatic controller will only control brightness and contrast for one spatial frequency. If successful there, the intent is to apply the theory to multiple frequency control if possible. This would give AMRL/HEA a convienient method of calibrating the generator at any time.

The computer recommended was the Mostek MDX Z80 based microprocessor. The microprocessor purchased was the Mostek MDX-CPU2 Z80 based Micro Computer Board which features six 24-pin memory sockets capable of accepting any combination of pin compatible RAM, ROM or EPROM. Two additional cards were purchased. One was the Mostek Parallel I/O Controller (MDX-PIO) card which contains four independent 8-bit I/O ports with two handshake or data transfer control lines. The other card purchased was the MDX-DEBUG Module which has 10K bytes of masked ROM which are filled with a Z80 Firmware Package. The Reticon Photodiode Array was a linear array of 1,024 photodiodes mounted on 25 micron centers. The diodes are stacked vertically; the width of each diode is 25 microns, the height 10 microns, and hence a spacing of 15 microns between each. The output of the Reticon circuit is a sample and hold boxcar signal which represents the level read by each diode. The converters to be used were an 8-bit D/A converter (DAC-UP8BM), a 12-bit D/A converter (DAC-HZ12BMR-1), and a 12-bit A/D converter (ADC-HX12BGC).

Problem Analysis

The basic purpose of my thesis was to program the Z80 computer to control brightness and contrast of the Optronix Signal Generator at a desired level. To do this the computer had to signal the Reticon Board to start a read of the luminance levels of the screen, then calculate the actual brightness and contrast of the screen, compare the actual values with the desired values, and output corrections to the generator. I programed the computer with a main driver routine which called different subroutines to preform the required operations. The various subroutines had to read in and store the screen luminance values, sort through the values to find the high and low values, calculate the average value of the stored readings, calculate the contrast value of the screen, and output correction signals for contrast and brightness. I also needed a routine to input the desired values for contrast and brightness. Additionally, I had to develop the circuits which would allow the two D/A converters and the A/D converter to interface to the computer. The A/D converter was used to change the luminance level read by the Reticon Board to a digital value. The D/A converters were used to change the correction signals from the computer into analog signals which were fed to the Optronix Generator.

The controller was, at conception, intended to control the generator output at any frequency. However, I determined that with the present equipment a controller for varying frequencies was impractical. My analysis in Appendix A shows that a minimum of 20 samples per cycle of a sinusoidal signal are required to insure obtaining the maximum and minimum values within five percent of their actual value (i.e. 95% of maximum). A time constraint is imposed by the Optronix Generator if one scan of the array is to be completed in one vertical trace of the Optronix screen. However, time constraints imposed by the computer did not allow that scan. The origional design called for every fifteenth diode of the Reticon Array Board to be stored by the computer (8). After programming the computer to read and store the values, I found that at least 10 micro-seconds were required to read and store each value. At this rate I needed about ten milli-seconds to scan the entire array. Since the vertical sync pulse of the generator occurs every 16.7 milli-seconds, and I wanted to complete a scan before the horizontal sync of the generator got to the position of the array, I had to change the origional timing requirement. Also, since I sampled every fifteenth diode, I had a maximum of 68 of the 1,024 diodes with which to calculate brightness and contrast of the screen. The diode array is approximately one inch long, therefore the highest frequency I could measure with the desired accuracy could only have three cycles per inch.

II <u>Design Procedure</u>

I divided the development of my thesis project into four steps. I covered the steps in four chapters although that was not the order in which I completed them. Chapter one covers testing of and the circuitry developed to interface the analog to digital (A/D) and digital to analog (D/A) converters to the computer. Chapter two covers checkout ofthe MDX computer cards, including the parallel I/O card. In chapter three I developed the assembly language program which was to input the desired brightness and contrast values, read the actual values off the CRT screen, and generate the corrected output signals. The purpose of chapter four was to tie all the developed systems together, developing the necessary circuitry to connect the Reticon Card and the D/A and A/D converters to the computer. The circuitry had to provide proper signals to allow the computer to be synchronized to the output of the Reticon card.

III Interface of Converters to The Computer

This chapter covers the development of the A/D and D/A converters with the additional circuitry required to interface them with the computer.

Port Select

Before I attached the converters to the computer data bus I had to develop a means of controlling the time at which data from the converters would be applied to the bus. To do this I designed a port select circuit to provide these trigger pulses. I selected two 74154 chips (13) which provide 16 seperate ports to be used for this purpose, letting one be an input chip and the other be the output chip. I used two because they were readily available and I was not sure how many pulses I would need. The two chips provided 16 input and 16 output pulses. With these ports, an IN or OUT command to the computer will provide the required start signal to the desired components. The circuit for the port select signals is shown in Figure 6 of Appendix B.

Converters

The A/D and D/A converters were wired according to the Datel Handouts provided and the Datel Handbook(1). The circuits were initially tested on the Elite 3 circuit design board. All three converters functioned , but it was necessary to debounce the switches used to provide the load pulses to the converters.

Twelve-Bit A/D Converter. The 12-bit A/D converter was wired as shown in Figure 7 of Appendix B for initial checkout. I used a variable power supply for the input signal and connected the output to the twelve lights on one of the Elite Design Boards. The start convert pulse was supplied by the clock on a DD-1 Design Board. The 12-bit A/D converter had a variance of about 4 bits with a steady DC voltage applied. I have assumed that this variance was because of variations in the power supply voltage since the converter was more accurate than the voltmeter I was using. After initial checkout the converter was connected to the computer and operation checked again. I used three 74125 tri-state buffer chips (13) to connect the 12-bit A/D converter to the data bus. I checked for correct operation when tied to the computer by applying a clock to the start convert input of the converter and reading the value input through ports 9 and 10 of the computer. Figure 8 of Appendix B shows how the tri-state circuit connects to the computer.

Eight-Bit D/A Converter. The 8-bit D/A converter was wired as shown in Figure 9 of Appendix B for initial checkout. I wired the input lines to the desired levels and measured the output with a voltmeter. The eight-bit converter had an internal latch circuit so after the initial checkout I connected the converter directly to the data bus. I used the output port select pulse from port 6 to control transfer of data from the computer. To complete checkout I sent an OUT (6) command to the computer and measured the output for the correct voltage.

Twelve-Bit D/A Converter. The 12-bit D/A converter was wired as shown in Figure 10 of Appendix B to make the initial checkout. Again I wired the input lines to the desired level and measured the output with a voltmeter. After initial checkout I used three 7475 data latch chips(13) to connect the 12-bit D/A converter to the data bus. I used output port selects for ports 3 and 4 to effect the transfer of data from the computer. Figure 11 of Appendix B gives an illustration of the latch circuit used.

LED Input Circuit

I decided to use an LED readout circuit to input the desired brightness and contrast values. The circuit display consisted of six Til-311 LED digits which provided two brightness digits and four contrast digits. The allowable range of brightness values was from 00 to 39,and the allowable range for contrast was from .0000 to .8192. The values are changed by pushbutton switches and they are input to the computer by an interrupt service routine when a push-button to the data strobe input of port F8 is depressed. Brightness can be changed in increments of 1 unit and contrast can be changed in .1, .01, or .00025 increments. The increment .00025 only registers as .0000, .0002, .0005, or as .0007 on the LED readout. The circuits used for the LED readouts of desired contrast and brightness are shown in Figures 12,13 and 14 of Appendix C. All digits but the least significant digit of contrast are generated with SN 7490 Decade Counter Chips(13). A 7473 J-K Flip-flop was used to generate the least significant contrast digit.

IV Checkout of The MDX Computer and Cards

Computer Checkout

I used the MDX DEBUG Card to check operation of the computer. With this card I could use the Debug Program to check any port and I was also able to make some programming checks. I preformed the operational checks outlined in the Debug Manual to insure the computer was operating properly. With the Debug Card I checked the operation of both the Parallel I/O card and the port select circuits which I had built to control operation of the D/A converters. An MDX power supply was purchased with the computer, and I used it to supply the +12 VDC, -12 VDC and +5 VDC power to the computer. The power supply was mounted in the bottom of a metal cage that I had made to house the computer and controller equipment.

Parallel I/O Card

I expected to use the parallel input/output card to input the desired brightness and contrast levels to the computer, so I wrote a short program with which I could check out the operation of the PIO card. The checkout program was loaded into RAM and with the aid of the DEBUG program I did verify proper operation of the card. I did have a few problems with initialization of the port and the proper strapping for each port, but they were in understanding the instructions rather than the instructions themselves.

Interrupt Checkout

I checked the interrupt circuit in the PIO Card with a small program which I loaded into RAM on the computer. I used the data strobe line of port F8 for the interrupt line and loaded port F9 of the PIO Card with the correct control signals for the interrupt vector and enable signals.

MDX Access Card

The Mother Board supplied with the computer was a fifty-six pin dual layer type, which requires a special connector that is no longer made. I needed some means of connecting to the address bus and the data bus, along with some of the other computer lines, so I designed an access card. This card brought the computer lines to two twenty-six pin connectors, which allowed me access to the buses and other computer lines. The access board drawing is shown in Appendix D.

V Assembly Language Program Development

Program design

The computer program was written in Z80 Assembly Language and tested on the Cromemco Z-2D Computer in the signal processing lab. Assembly errors were found and eliminated using the Macro 80 Program. The program was written with the aid of the Wordstar Editor, and debugged using the Cromemco Trace Program. The assembly language nemonics are listed in Z-80 reference handbooks (5,14). I made no effort to minimize the amount of code used since I was unfamiliar with Z80 Assembly Language. The object of the program was to have the Reticon Photodiode Array Board read and store several screen luminance values, and then use the computer to calculate the actual brightness and contrast from the screen. The actual values would then be compared to the desired values and the proper corrections made. The desired brightness and contrast values are input to the computer by an interrupt routine which is called through one of the parallel I/O ports. To maintain top down structured programming practices, I broke the program down into several smaller routines. I tried to make a separate subroutine for each of the functions preformed by the computer.

Main Driver. I wrote one major driver program to control the sequence of events. The main driver then calls other subroutines in order to read the values and make the calculations and corrections. The first portion of the main driver program initializes the ports of the parallel input/output card to input, even though the default values are input. One port (F8) is also initialized to provide the low byte of the interrupt vector for the interrupt service routine. The I register is initialized to the high byte of the interrupt routine. I also set an initial value for contrast and brightness, both desired and output, at this time. The output values are the values which are sent to the generator to change contrast or brightness.

The next section of the program is actually where the main driver block begins. In this section 50 (hexadecimal) memory locations are set to zero in preparation of the Reticon Card read. This step is probablly not necessary, but I felt better starting a scan with nothing in memory. The driver then calls, in order, the read array routine, the sort routine, the averaging routine, and the calculation and correction routines. After calling each routine, the driver then returns to the memory zero portion and repeats the process. The main block repeats continuously with the only deviation being when new desired values for contrast and brightness are input by the interrupt service routine.

Read The Array Routine. The first routine called by the driver program is the routine to read in the actual luminance values detected by the photodiode array. The values are stored in sequential locations starting at location RANDOM (F850H). Two locations are required for each twelve bit value stored. This routine also tracks the number of values stored and that number is stored in COUNT. The number of values expected to be stored is 68, but I still tracked and stored this value in case of future design changes.

Sort Values Routine. The second routine searches through the stored values to find the highest and lowest value. These are then stored in locations MAX and MIN respectively. This routine starts comparing numbers at RANDOM and continues until COUNT numbers have been compared. The comparison starts on the most significant four bits, and if no match is found there, goes to the least significant eight bits.

Average Routine. The third routine finds the average value of all the values which were stored. This 12-bit value will be stored in location AVE. Again the routine will start at RANDOM and average COUNT numbers. The routine keeps a running average as it checks each number, updating the average on each pass. The values read and calculated to this point of the program have been 12-bit values.

Brightness Routine. The next routine takes the average value that was computed in the averaging routine and converts that value to an 8-bit value by rotating it left four times. This rotation is done through two registers to maintain the correct value. The 8-bit value is then compared with the desired brightness and a correction for brightness is output to the Optronix Generator. The correction is added to the previous output value, which was stored in BRITE, and the new value is output and stored in BRITE.

Contrast Routine. The routine to make the contrast correction must first calculate the actual contrast on the screen. I did that by dividing the difference between the maximum and minimum values by twice the brightness value. This calculation is based on the following formula:

$$\text{Contrast} = (B\ \text{max} - B\ \text{min})/(B\ \text{max} + B\ \text{min}) \qquad (1)$$

Where; B max = Maximum screen luminance value

B min = Minimum screen luminance value

The calculated value for contrast is then compared to the desired contrast value to find the correction signal. Again the difference is added to the previous output and the new signal sent to the generator.

Return. At this point the program will return to the initialization routine and zero the locations for the new set of values to be read in. The interrupt capability will be enabled during the initialization, sort and averaging routines and will be disabled during the read array and correction routines. The interrupt will also be disabled initially until after the initialization of ports and interrupt vectors.

Interrupt Routine. The interrupt routine inputs a two digit value for brightness and a four digit value for contrast. The values are input in binary coded decimal form and converted to their binary equivalent by the routine. After being converted to binary they will be stored as the desired contrast and brightness values before returning to the main program.

Contrast is converted directly to a binary number, but since the CRT screen brightness voltage is not a linear function of the screen luminance or brightness, it was necessary to store a table of values to convert the desired brightness level into a binary value which represented the desired voltage level. For purposes of this thesis, contrast was considered to be linear for small areas around the average brightness of the screen. In order to determine the table values for brightness, screen brightness had to be measured with a light meter and that value compared to the voltage level which produced that luminance. I then had to determine the number of bits required by the 8-bit D/A converter to produce that voltage. This was the value to be stored for that particular brightness or luminance value.

The desired brightness and contrast values are input by activating the interrupt routine with a pushbutton switch which the operator can depress. The switch activates an interrupt signal by causing the data strobe line of port F8 to go low. This calls an interrupt service routine to input and store the values. The range of brightness is from 0 to 39 in unit increments. The range of contrast is from .0000 to .8192 in increments of approximately .00025 units. Only four digits are displayed, the least of which has only four values, 0,2, 5,and 7. The value for contrast is converted to a 13-bit binary number then divided by two to reduce it to twelve bits. Brightness is converted to an 8-bit binary number then the input value is converted to a corrected brightness value through a table which is stored in ROM. The desired decimal values are input through the parallel input/output card (PIO) supplied with the computer. The PIO card provides the interrupt signal and the low byte of the interrupt vector address. The PIO card causes the computer to jump to the correct location for the input service routine when the interrupt is enabled and the data strobe line is pulsed low. A copy of the program and the flowchart is contained in Appendix E.

VI Connecting Systems Together

Power Requirements

The MDX Computer had a power supply with it that was purchased with the computer. This power supply was capable of supplying the +12, -12 and +5 VDC power requirements of the computer and the other MDX Cards, but the converters and the Reticon Card required +15 and -15 VDC for operation. Looking at the requirements for the computer system and the other systems, I found that another power supply was needed. For temporary use on the project I used a Powertec Power Supply from the AFIT lab inventory with the knowledge that if everything worked I had to find another power supply before the project would be complete.

Interconnection

The Block Diagram in Figure 1 shows how the systems are connected together. In this diagram I have shown the Reticon Card, the MDX Computer, and both of the boards which I wire wrapped the additional circuits on. A board diagram of each of the wire wrapped boards is shown in Appendix G. I numbered each of the Integrated Circuits on each board for easier identification. I tried to retain the chip number on each of the other drawings, in order to help keep track of their loocations. The boards which I called Wire Wrap Board 1 and 2 contain the circuits which interface the components together. Wire Wrap Board 1 contains the converters along

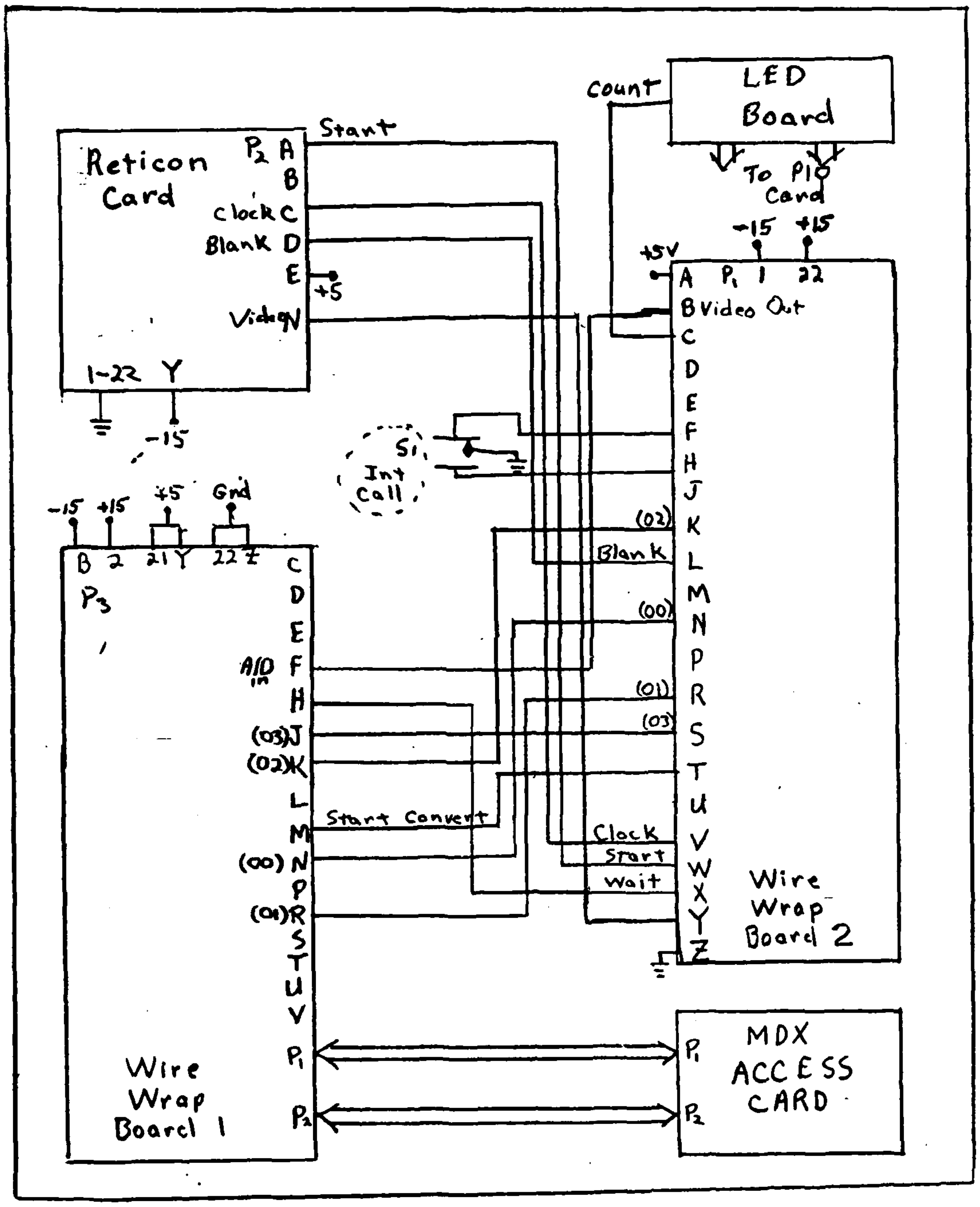


Figure 1. Block Diagram of The System Interconnections

with the tri-state and latch circuits which connect them to the computer. That board also has the port select circuit on it. Wire Wrap Board 2 contains the modulo fifteen counter which controls the selection of a diode value of the Reticon Array to be stored, and the flag generation circuits which tell the computer the status of a scan. This board also has the sample and hold circuit which was used in the process of selecting a diode value to be stored. The timing diagram in Figure 2 shows the timing of a start signal and how it is synchronized. The timing diagram also shows the selection pulse which is used to store every fifteenth value of a scan.

Counter

To have the computer read and store every fifteenth diode of the Reticon Card, I needed some means of selecting the value to be stored. I designed a modulo 15 counter to provide a sync pulse which could select every fifteenth diode to be read. The design of the counter was completed using standard design techniques (7,11). I used four J K Flip-flops for the counter and the circuit design is shown in Figure 10 of Appendix F. The counter is active only while the Reticon Blanking Pulse is low. I used two output pulses from the counter to control the passing of values to the computer while the counter is active. I let the thirteenth count pulse the sample and hold circuit and then the fourteenth count pulse the A/D converter to change the value to digital. The counter resets on the fifteenth count. The internal

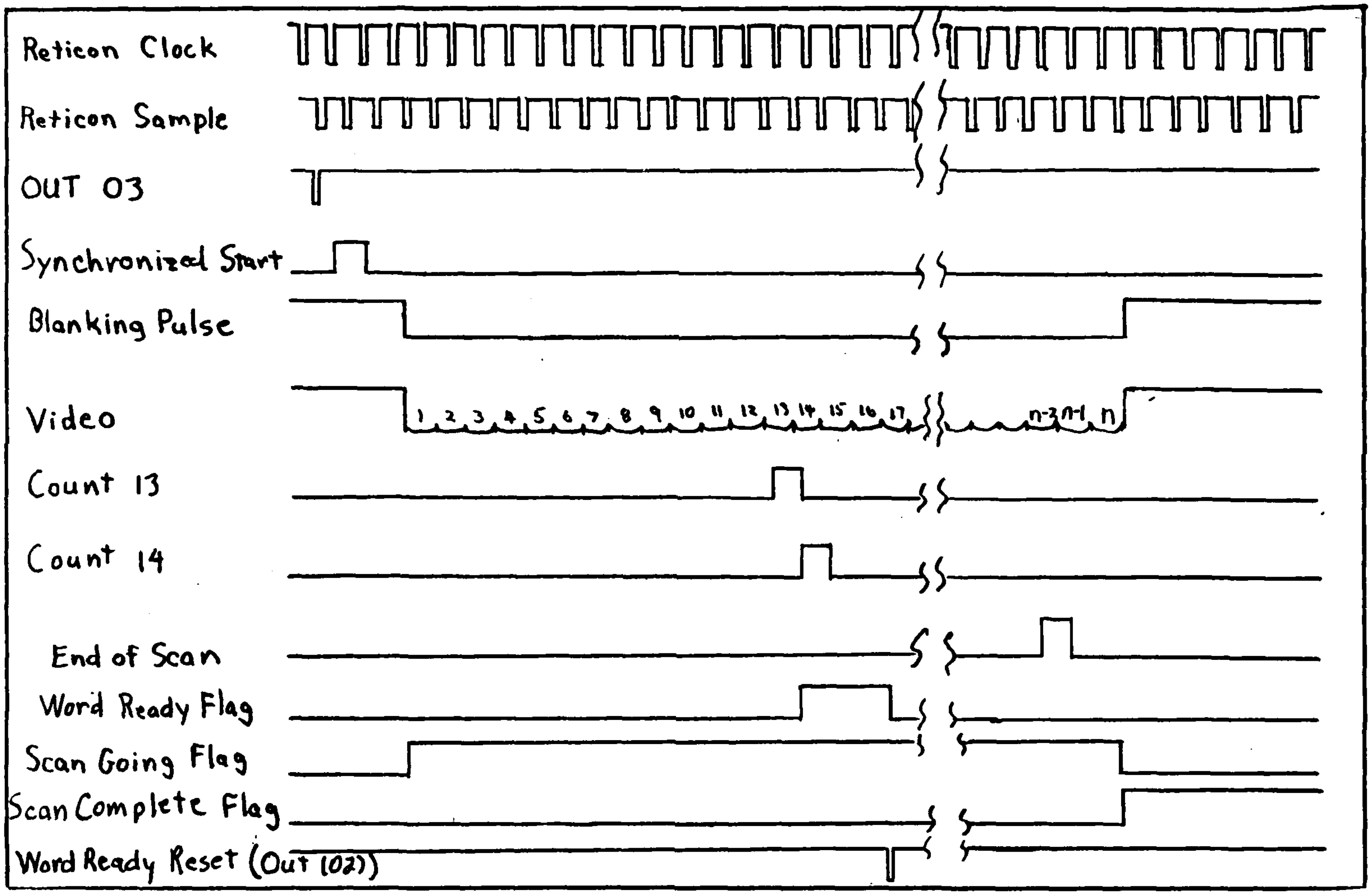


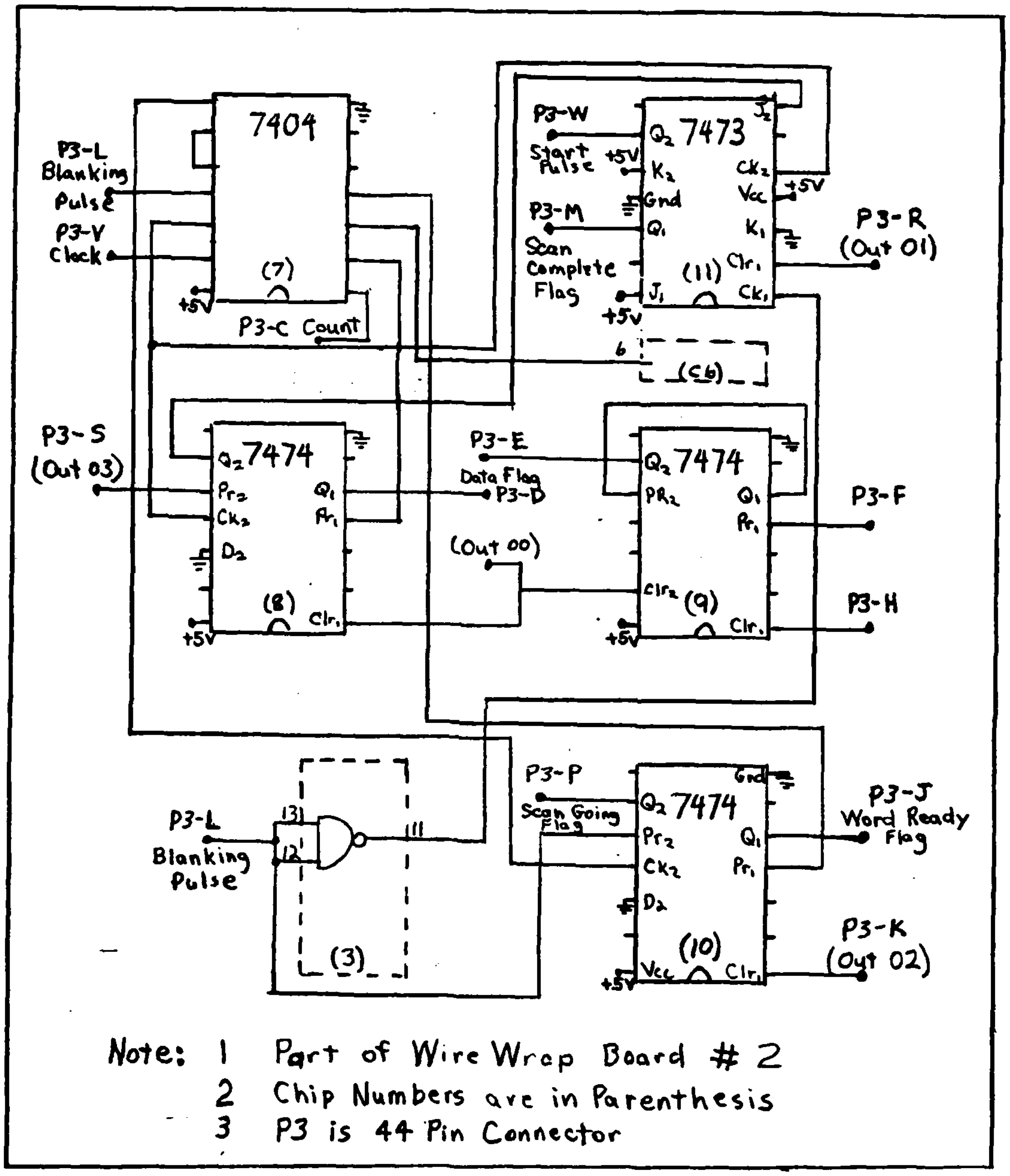
Figure 2. Timing Diagram of Scan Signals and Flags

clock pulse from the Reticon Card was used while the blanking pulse is low to strobe the counter. This synchronized the output pulses with the values being read. A diode of the array is read on each clock pulse during a scan, hence the output pulses select one of every fifteen values read to be stored. The output generated on the thirteenth and fourteenth counts go to the sample and hold circuit and to the A/D Converter Circuit.

Flag Generation Circuit

In order to synchronize the computer to the array scan, I had to develop a means of telling the computer the status of a scan. I decided to use flags to talk to the computer and to have the computer test flag status before conducting an operation. I used this method because there was not much time to spare between the scan of every fifteenth diode and I wanted to be sure the computer was ready to store a value as soon as the value was ready. Firgure 3 shows the circuits which generate all the flags used.

I used the OUT (03),A command to generate the start signal for the Reticon Card. To synchronize the start signal to the Reticon Internal Clock I conditioned it with two flip-flops and the internal clock. Figure 2 shows the timing sequence of the start signal. I used the blanking pulse from the Reticon Card to start the modulo fifteen counter. The counter then provided the pulses which would select every fifteenth diode value to be stored.



Note: 1 Part of Wire Wrap Board # 2
2 Chip Numbers are in Parenthesis
3 P3 is 44 Pin Connector

Figure 3. Flag Generating circuit

After starting a scan, the computer checks the Word Ready flag condition. The computer continues testing until a value is ready. The fourteenth count of the counter is used to set the Word Ready flag. The flag is reset by an OUT (2),A command from the computer after the value has been stored. The fourteenth count is also used to generate a wait to the computer to prevent a check and reset while the count is still at fourteen.

The sequence of storing values continues until the blanking pulse from the Reticon Card sets the End of Scan flag. The computer checks for the end of scan after each value is stored, returning from the Read Array subroutine when the flag is set. The OUT (01),A command from the computer resets the End of Scan flag before returning.

During the array scan, the blanking pulse generates a flag to indicate that a scan is in progress. I also have flags to indicate that the desired values have been changed without inputting them into the computer and a flag to indicate that the data strobe line was activated. These are both reset by the OUT (00),A command during the Interrupt Service Routine.

Reticon Board

The reticon circuit board required an external start signals to begin a scan. This was provided by using an the output port select pulse from port 03. To generate a start signal synchronized with the internal clock of the Reticon Card clock. This pulse started the scan of the 1024 diodes and every fifteenth was stored by the computer. The out 03 pulse was synchronized with the Reticon Board Clock using two flip-flops and then applied directly to the external start input of the card.

I also used flip-flops to generate the flags required to tell the computer the status of the scan card. The blanking pulse from the Reticon card was used to tell the computer that the last diode of the card had been read. A 7473 flip-flop was used as the flag to tell the computer a scan was in progress. The 7473 is set when the scan begins and reset by the positive transition of the blanking pulse. The output of this flip-flop was used as a flag to inform the computer when the scan was complete. This flag was input through bit 01 of port FC of the Parallel I/O controller. The modulo 15 counter was used to signal the computer when a value of the photodiode had been sampled by the sample and hold chip and was ready to be input. Count 14 of the counter generated the pulse required to set the value ready flag. This flag was input through bit 00 of port FC. Count 13 of the counter generated the pulse to allow the sample and hold chip to sample one of the diode values. I also use this pulse to take the WAIT line of the computer low, preventing a reset of the word ready flag prior to completion of the pulse. The counter is only active while the blanking pulse is low. This is accomplished by logically anding the clock out from the Reticon card with the inverted blanking output from the Reticon card.

Sample and Hold Circuit

The A/D Converter was not fast enough to convert every value that the diode array provided, so an extra sample and hold circuit was used to hold the selected value long enough for the converter to convert it to binary.

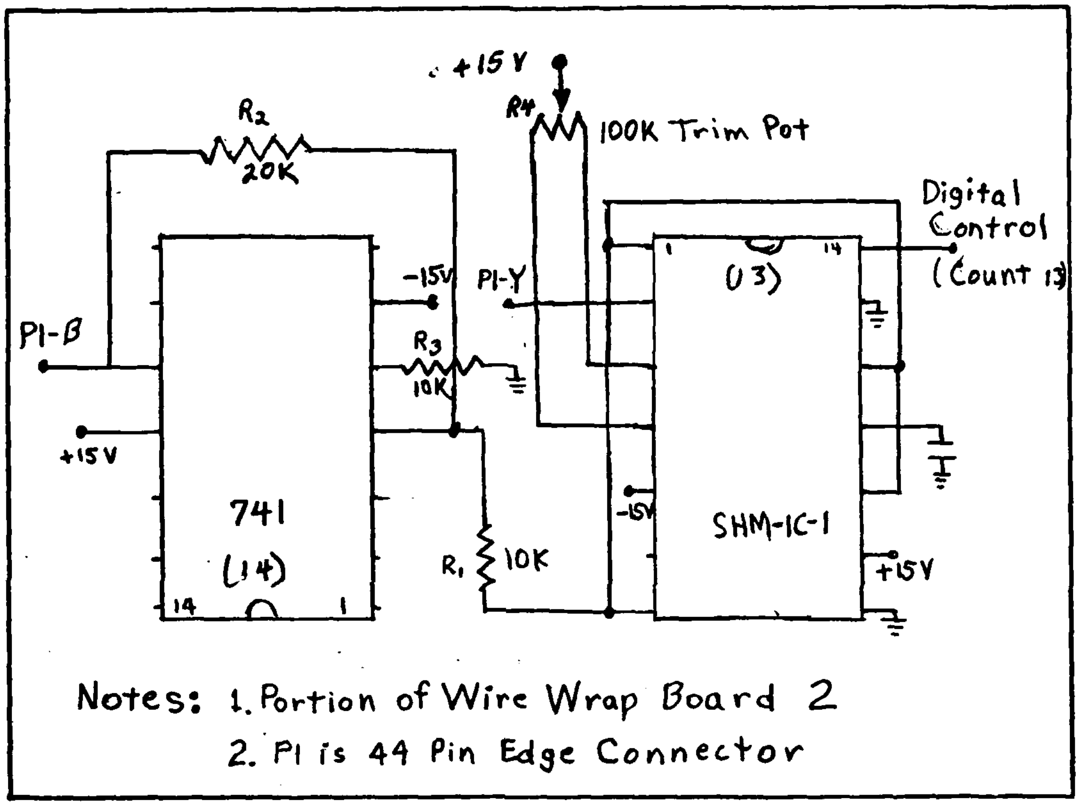

+15V
R4
100K Trim Pot
R2
20K
Digital Control (Count 13)
-15V
P1-Y
(13)
14
P1-β
R3
10K
+15V
741
(14)
14
R1
10K
-15V
SHM-IC-1
+15V
Notes: 1. Portion of Wire Wrap Board 2
2. P1 is 44 Pin Edge Connector

Figure 4. Sample And Hold Circuit

Figure 4 shows the diagram of the sample and hold circuit. The sample and hold chip held the thirteenth diode value and the 741 operational amplifier inverted the value and scaled it to ten volts maximum. The thirteenth count of the counter was used for the digital switch to the sample and hold chip, so the thirteenth value was actually the value which was stored. Count fourteen was then applied to the A/D Converter to cause the conversion.

Luminance Processor

A luminance processor was required to control the contrast of the video signal to the screen. A suitable processor to do this is the CA3144G TV Luminance Processor which is made by RCA. Actually a substitute processor was

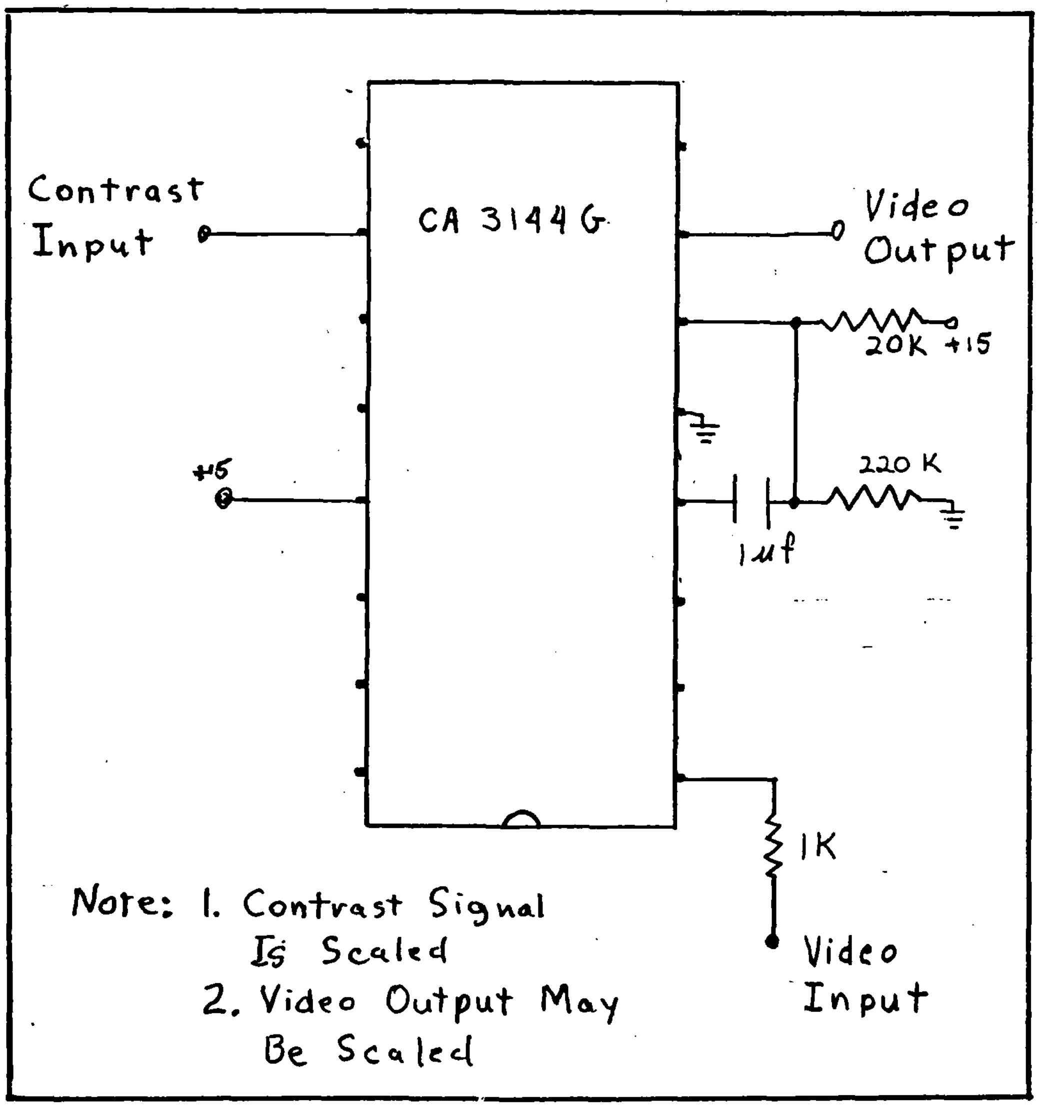
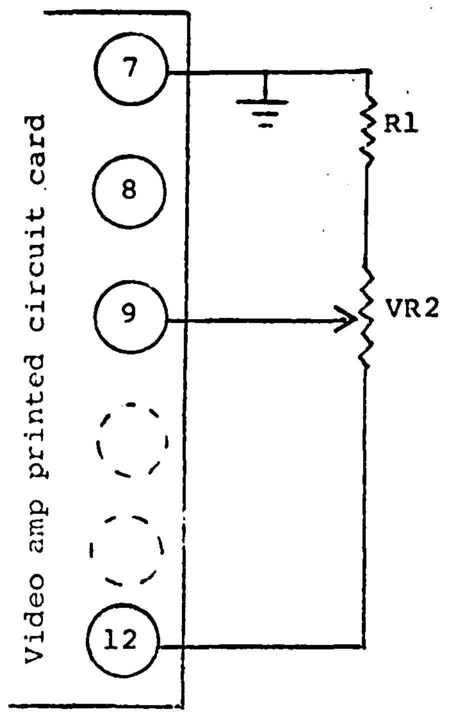
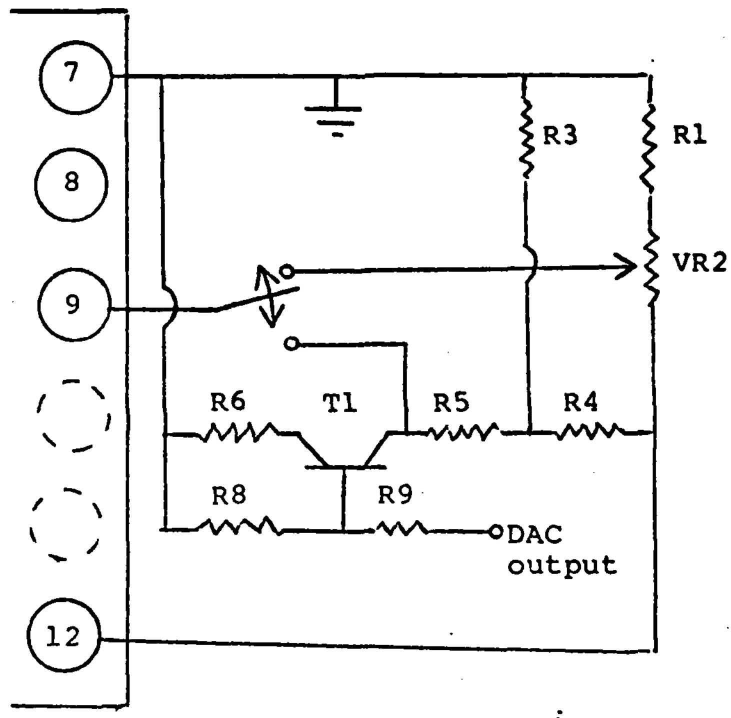


Figure 5. Luminance Processor Circuit



a) Present Wiring

R1	=	47K
VR2	=	100K
R3	=	47K
R4	=	18K
R5	=	120K
R6	=	10K
R7	=	270
R8	=	100
T1	=	SK 3024



b) Modification of Wiring Allowing Digital or Front Panel Control

Figure 6. Brightness Voltage Input Location

used, Sylvania ECG 985, but the processors are the same. The processor circuit is shown in Figure 5. The contrast correction signal from the computer is fed to the contrast control of the processor to control contrast. The processor also has a brightness control, but the previous study found that more control of brightness was required, as shown in Figure 6, because of the normal DC restoration function typical of contemporary CRT monitors. brightness. This part of the development was not actually conducted due to the bad A/D Converter chip, but the design was carried forward from the design which was developed in the first thesis. Testing will be accomplished if the converter arrives before I depart the area.

VI System Operation

The system operates automatically, controlling both brightness and contrast at the level specified by the user. The main driver program sets in an initial value for the desired values of both contrast and brightness when first started, but as soon as the data strobe line goes low to call the interrupt routine these desired values are changed. The operator can change either the desired brightness or the desired contrast by setting the LED values to the new value and pushing the Set switch on the console. This switch is a pushbutton switch that causes the data strobe line of parallel port F9 to go low. This sets the computer interrupt flag, and as soon as the interrupt is enabled, the computer will store the new desired values. I have a flag that, at present, is connected to a light to indicate that the desired values have been changed and the desired values have not been updated. This flag is set any time that the operator changes any of the LED display outputs and remains set until the computer goes through the interrupt service routine. When the Data Change Flag is low the operator is assured that the value shown on the LED display is the value the computer has.

The system operates by a Main Driver Program which controls the sequence of events that occur except for the interrupt. The Main Driver Program sets the initial values for contrast and brightness, and then it puts zeros in

80 (hexadecimal) locations of memory. The remainder of the events are controlled by subroutine calls from the main driver. The subroutines control the reading of luminance values from the screen, finding the average value, calculating contrast, and outputting the correction signals to the generator.

The twelve-bit A/D converter changes the video values from the Reticon Card into digital values before they are stored by the computer. The twelve-bit converter changes the contrast value into an analog value, which is then used to control the contrast signal. The eight-bit converter changes the brightness value to analog. The luminance processor circuit takes the video signal from the generator and uses the brightness and contrast signals to control those levels. A diagram of the luminance processor circuit was shown in Figure 4. Although this circuit has not been checked because I did not have the twelve bit A/D converter, the previous work on this project had determined that the processor did control brightness and contrast of a video signal.

Although I never actually used system to control the Optronix generator, I believe the system will function. All of the modules have been checked separately and seem to function well. The computer program has been checked with the Cromemco Emulator in the Signal Processing Lab at AFIT and the actual system does interface with the computer. Once the A/D Converter arrives it should be a simple matter to finish the project. All that remains is insuring that the system functions and then mounting all the lights and switches on the console which will hold the controller circuit. I have left provisions for switches to be used to input desired values and also a switch to cause a system interrupt to input the desired changes in contrast and brightness. The LED Input Circuit has not been manufactured yet, but the protype is still mounted on the Elite Breadboard and can be used temporially. The switches still have to be mounted on the console, and the wire wrap boards and the Reticon Board have to be mounted in the circuit box provided.

VIII Problem Areas of Design

Reticon Problem

After the initial design phase was completed, I encountered one major design problem in the process of checking the operation of the system.. In the design I used a portıon of the Reticon Board Circuitry to control the flags that told the computer the status of a scan. The circuit I used was the blanking pulse which goes active low during the scan of the array. The Reticon Circuit came as two separate circuit cards which my predecessor had connected with a ribbon cable about three feet long. The major reason for the long cable was to allow placing the diode, which was separate from the majority of the circuitry, directly on the screen of the generator. In trying to operate the Reticon Card with the computer I found that the blanking pulse from the Reticon Card was low at all times. After a considerable amount of trouble shooting I discovered that the ribbon cable was causing a noise problem with the end of scan pulse generated in the diode. This caused the blanking pulse to remain low, since the end of scan pulse was used to bring the blanking signal high again. When I connected the array board directly to the Reticon Drive Card, I had the proper blanking pulse.

A/D Converter Problem

After I seemed to have worked out all of the minor problems of interface between the Rectcon Card and the computer, I attempted to preform the scan operation and store the values as they were scanned. I did this by borrowing a RAM card and moving the program into a location in RAM where I could set a breakpoint immediately after the scan operation. This would allow me to check how the store routine was working by checking the values stored. The first time through the Read Array Routine the computer stored 68 values. All of the stored values were the same, which at the time I thought strange but not unreasonable. I varied the amount of light on the diode array and tried again. I found the same 68 values were stored. I then checked the A/D Converter and found that the internal clock of the converter was not operating. At some time between the initial checkout of the converter and my final checkout, something had boiled on the converter. A new converter was ordered and meanwhile I tried to rig up a substitute converter in its place. The only converter available here with operating directions was an ADC 0816. I connected the converter as shown in the available handout and made an initial check. The converter worked initially with a 100KHz clock applied and a random convert signal sent manually. However, when I attempted to pulse it with the convert signal generated by the modulo 15 counter, I found that the conversion was no longer accurate. This is as far as I have gotten in the project.

Generator Hookup

To prepare for the acquisition of a good converter I began the steps to connect the controlling circuitry to the generator. This was a matter of finding where the brightness and contrast signals could be inserted into the generator to produce the desired results. The first step was to find the voltage levels there, and then determine how to scale the brightness and contrast signals to the proper levels. Once this was done it was a simple matter to scale the signals and send them to the generator.

IX Conclusions

This project has shown that an automatic controller for the Optronix Signal Generator is feasible. Although the controller is not functioning, the basic design has been tested to the point of applying the actual feedback control. The computer program for the controler was much shorter than expected and appears to be adequate.

The Reticon Array is capable of reading a video signal from a CRT screen. That was done with a sine-wave grating signal with the result being a sinusoidal video being output from the Reticon card.

The MDX Computer is very flexible and can be expanded to include many functions not looked into in this project. One possible area for expansion is expansion of the controller to multiple frequency control

X Recommendations

There does not seem to be suffiecient time to complete this project during the current thesis effort. I do not think that completion will be a major task. All that remains is to connect the controlling circuitry to the generator. All checks to this point have shown that all the systems except the A/D Converter work. When the converter arrives I only expect to have to place it in the circuit and connect it to the generator. Once the circuitry is connected I expect only to have to change the contrast and brightness values on the LED circuit and input them by the Insert Switch in order to control screen brightness to any desired value. If there are further problems encountered they will probably be in the area of the digital control of the brightness and contrast signals.

BIBLIOGRAPHY

1. Datel Intersil. Engineering Product Handbook. Datel Systems Inc., 1979.

2. Ginsburg, Arthur P. et al. "Suprathreshold Processing of Complex Visual Stimuli: Evidence for Linearity in Contrast Perception," Science, 208: 619-621 (9 May 1980).

3. Hall, John A. Jr. and John J. McCann. "Effects of Average-Luminance Surround-on the Visibility of Sine-Wave Gratings," Optical Society of America Journal,70 (2): 212-219 (February 1980).

4. Kabrisky, Matthew. Lecture materials presented in EE 6.17, Information Processing In Nervous Systems. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, 1980.

5. Leventhal, Lance A. Z80 Assembly Language Programming. CA: Osborne & Associates, Inc., 1975.

6. Lovegrove, W.J.,et al. "Specific Reading Disability: Differences in Contrast Sensitivity as a Function of Spatial Frequency," Science, 210: 439-440 (24 October 1980).

7. Mano, M. Morris. Digital Logic and Computer Design. New Jersey: Prentice-Hall, Inc., 1979.

8. Martindale, Kenneth L. Programmed Control of Optical Grating Scales for Visual Testing. MS thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, December 1980. ().

9. Mostek Corporation. Mostek 1979 Microcomputer Databook. Mostek Corporation, 1979.

10. Nagano, Takashi. "Temporal Sensitivity of the Human Visual System to Sinusoidal Gratings," Optical Society of America Journal,70 (6): 711-716 (June 1980).

11. Rhyne, V. Thomas. Fundamentals of Digital Systems Design. New Jersey: Prentice-Hall, Inc.,1973.

12. Sekuler, Robert, et al. "Human Aging and Spatial Vision," Science, 209: 1255-1256 (12 September 1980).

13. Texas Instruments Incorporated. The TTL Data Book(Second Edition). Texas Instruments Incorporated, 1976.

14. Wadsworth, Nat. Z80 Instruction Handbook. CT: SCELBI Computer Consulting, Inc., 1978.

Appendix A

Analysis of Required Signal Sample Frequency

Analysis of Required Signal Sample Frequency

The signal being measured on the CRT screen is a sinusoidal grating signal which varies in brightness in a sinusoidal manner as a function of distance across the CRT. The measurements taken from the screen by the Reticon Board are used to calculate the actual brightness and contrast. Since contrast is calculated from the maximum and minimum values read, the accuracy of the measurement will determine how accurate the calculated value for contrast is. I want to measure the maximum and minimum values of the signal as close as possible. Since the signal is sinusoidal, and the maximum value of a sine-wave is one, I can take the inverse sine of the percentage of accuracy I want and find the number of measurements required to obtain that accuracy.

I chose to have the maximum and minimum values read to within 5% of their values. With this selection I can take the inverse sine of .95 to determine how many samples of each cycle I need. This number gives me the largest angle division of a 360 degree arc which can be used. Since there are two pi radians in a circle, I divide two pi by the inverse sine of .95 to find the number of measurements which are required from one cycle of the signal. Equations one through six show this calculation.

$$\text{360 degrees} = \text{2pi radians} \tag{1}$$

$$\text{pi} = \text{3.1416 radians} \tag{2}$$

inverse sine (1) = 1.571 (3)

inverese sine (.95) = 1.253 (4)

1.571 - 1.253 = .318 radians (5)

2pi / .318 = 19.76 (measurements) (6)

When rounded off, the calculated number of required measurements is 10 but I used 20 to give an added measure of accuracy.

APPENDIX B

A/D and D/A Converter Circuit Layout

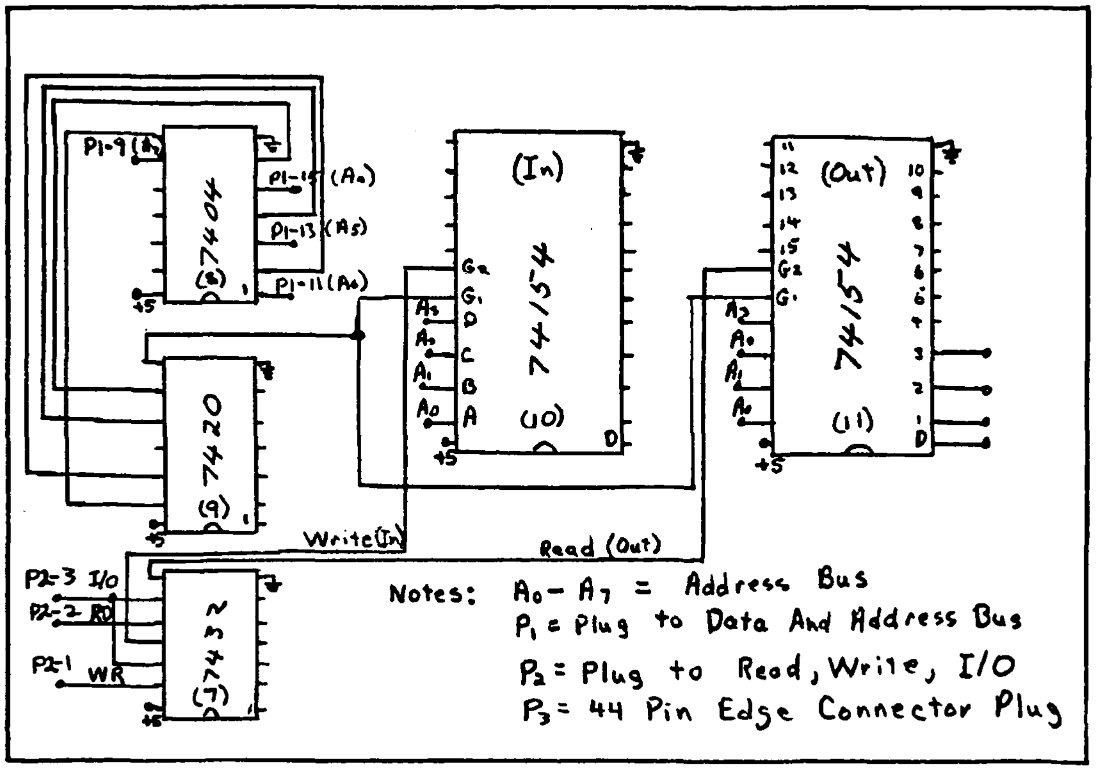


Figure 7. Port Select Signal Circuit

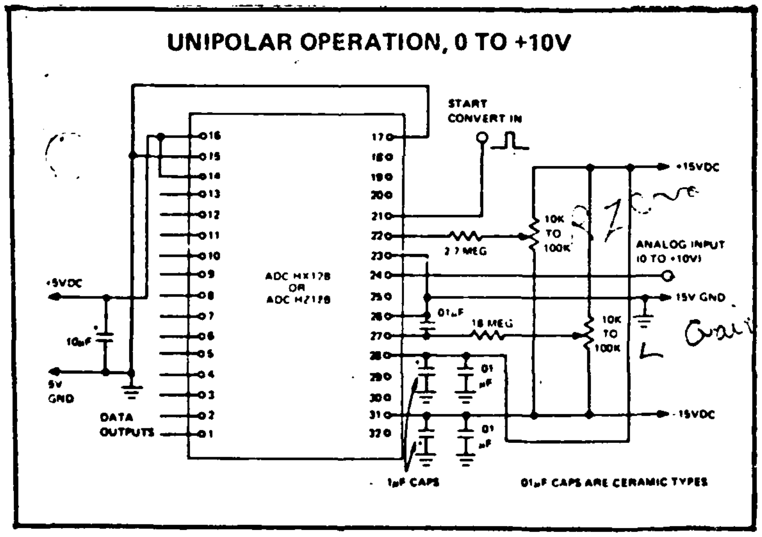
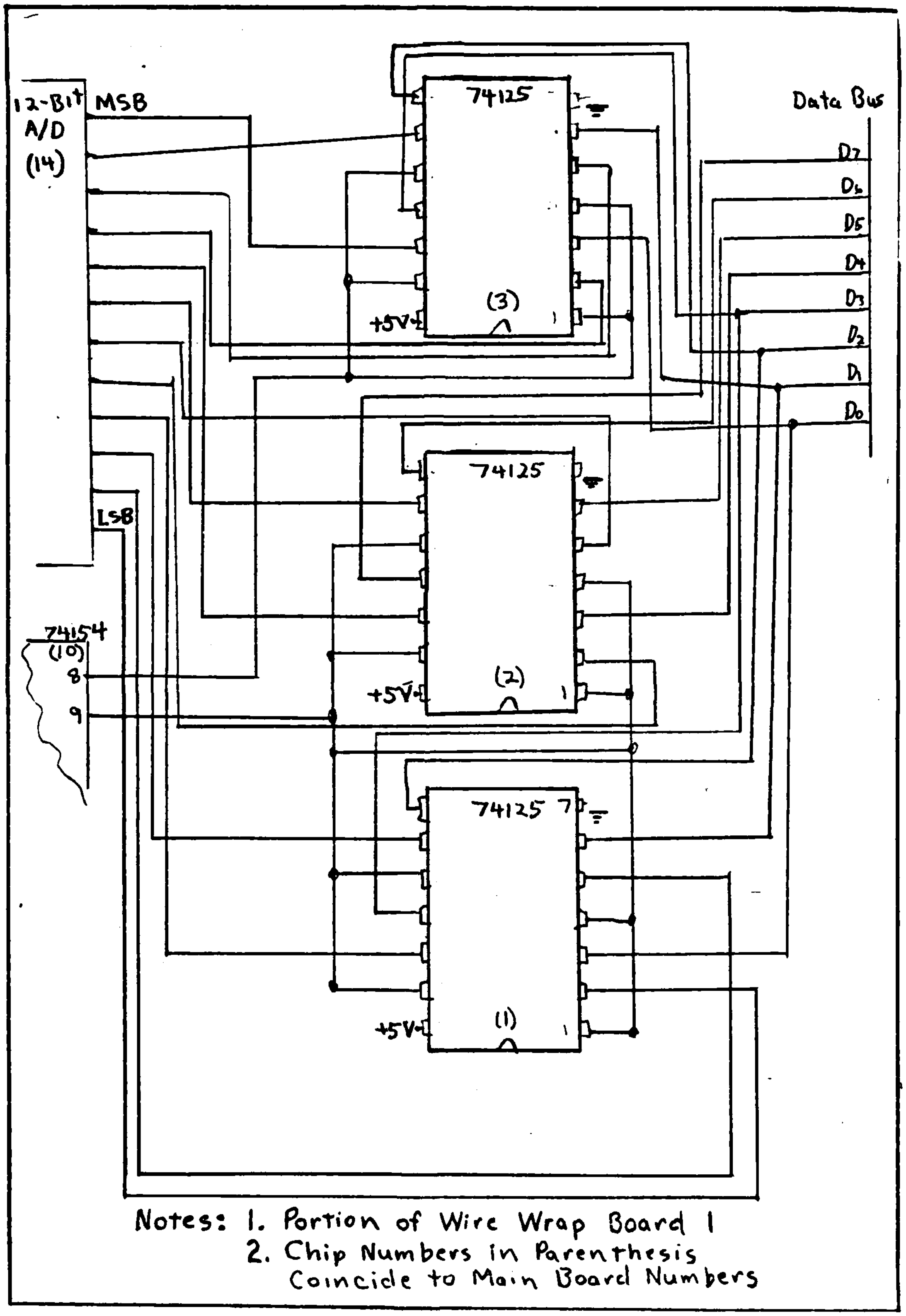


Figure 8. Twelve-Bit A/D Converter Circuit



12-Bit A/D (14)
MSB
LSB
74125
(3)
(2)
(1)
+5V
Data Bus
D7
D6
D5
D4
D3
D2
D1
D0
74154 (10)
8
9
Notes: 1. Portion of Wire Wrap Board 1
2. Chip Numbers in Parenthesis Coincide to Main Board Numbers

Figure 9. Tri-State Buffer Connection To Data Bus

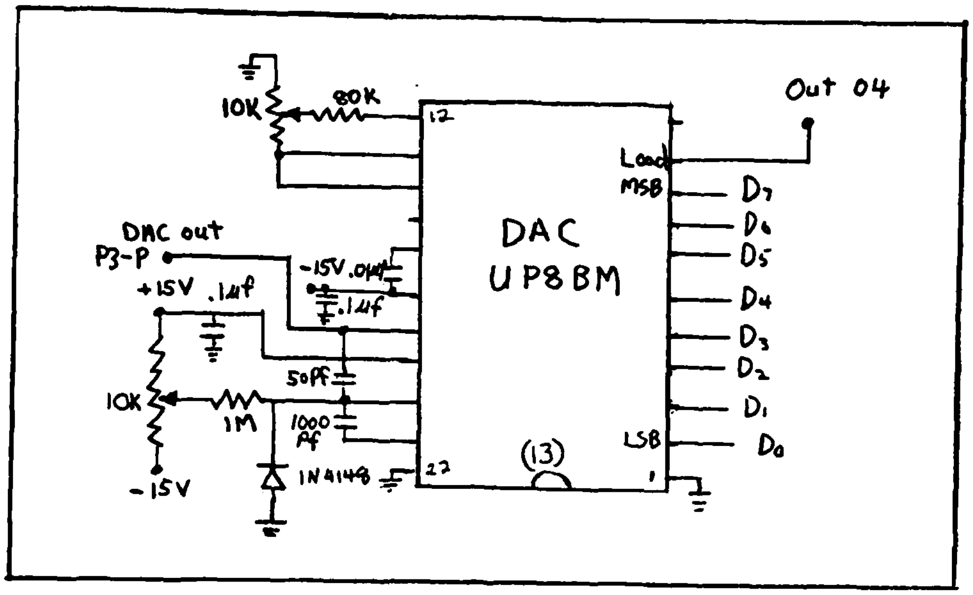


Figure 10. Eight-Bit D/A Converter Circuit

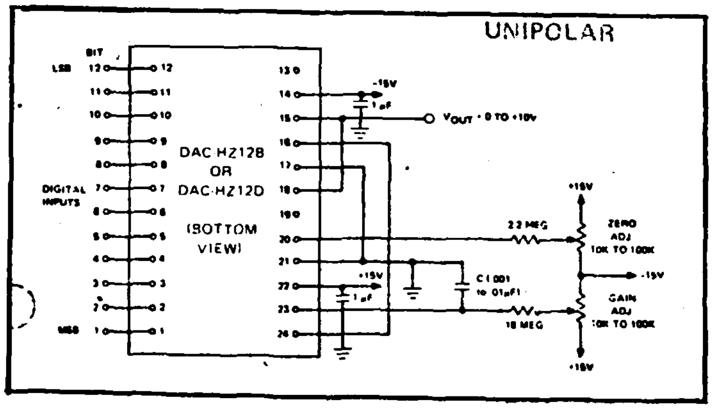
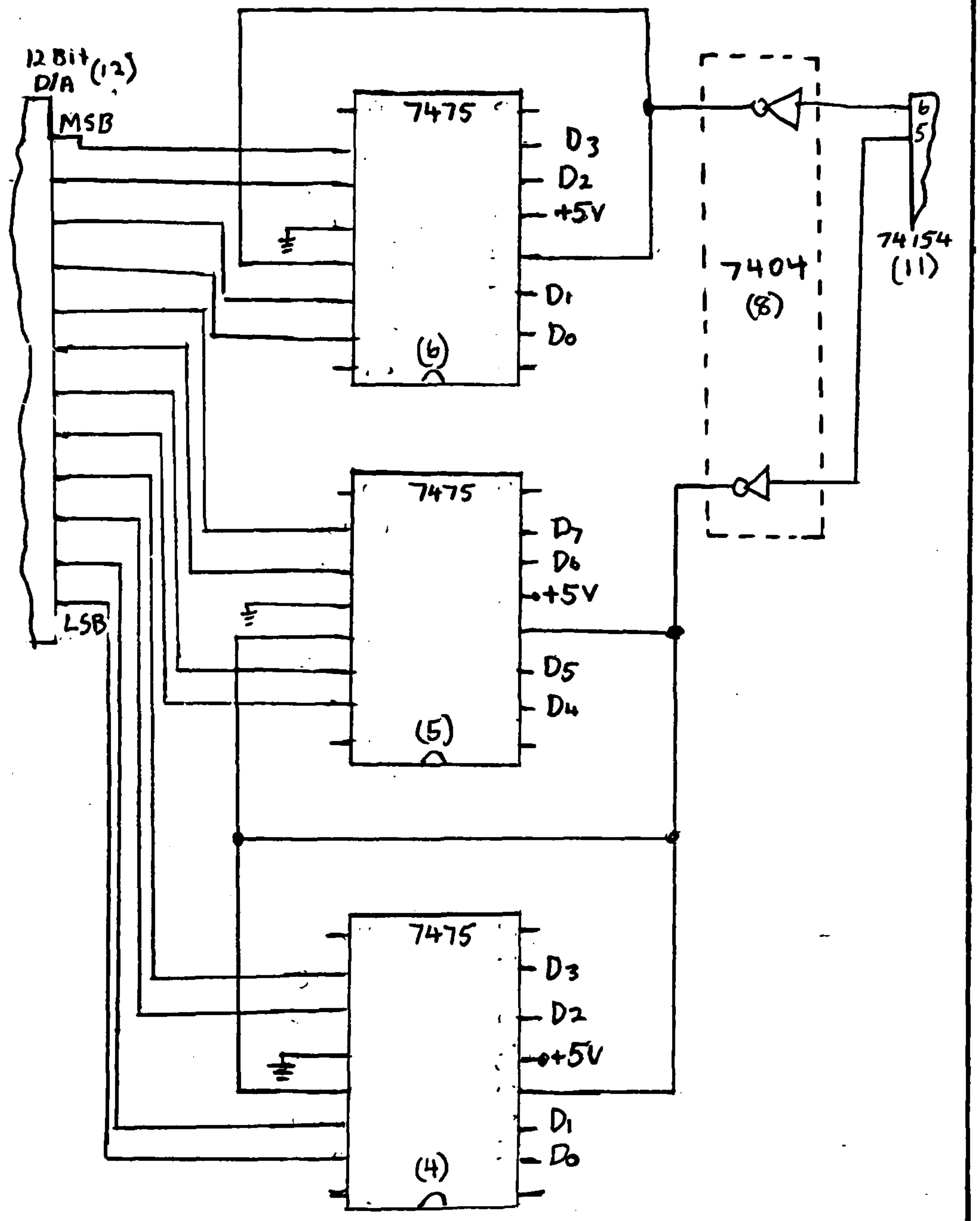


Figure 11. Twelve-Bit D/A Converter Circuit



Notes: 1. Portion of Wire Wrap Board 1
2. Numbers in Parenthesis Coincide To Chip Numbers On Main Board

Figure 12. Latch Circuit Connection To Data Bus

APPENDIX C

LED Input of Circuit Design Procedure

LED Input Circuit Design Procedure

The desired values for brightness and contrast are input to the computer by an interrupt service routine which is called when the data strobe line of port F8 of the parallel controller is pulsed. Each digit is generatea by a counter and displayed to the operator by a Til-311 LED display.

Brightness

Desired values for brightness range from 0 to 39 lumens at intervals of 1 lumen. The values are strobed by a pushbutton switch which operates one half of a 7474 flip-flop to debounce the pulses. Two decade counters (7490) represent the digits and are connected as shown in Figure 13.

Contrast

The desired contrast values range from .0000 to .8192 in increments of .00025. The least digit only varies through 0, 2, 5 and 7. Although this is the least digit displayed, the actual values which are represented by the display are .00000, .00025, .00050 and .00075. The other three digits displayed can vary from 0 to 9. The greatest two digits are strobed by their own pushbutton and 7474, while the last two digits are pulsed by a single pushbutton and 7474. Figure 14 shows the design of this circuit.

Reset

Both brightness and contrast are reset by a reset line to prevent them from going above the maximum allowable values. The design of the contrast reset circuit is shown in Figure 15. The reset for brightness is included in Figure 13. Both reset circuits are designed to only handle numbers over the desired maximum as the value approaches that maximum. I have noticed that at start up the values sometimes do have readings in the hexidecimal range. This will be no problem for the computer since the number will just be converted to binary, but it may be confusing to the operator. If this occurs, the values should be changed until decimal values are shown. That will eliminate any misunderstanding of a hexidecimal number.

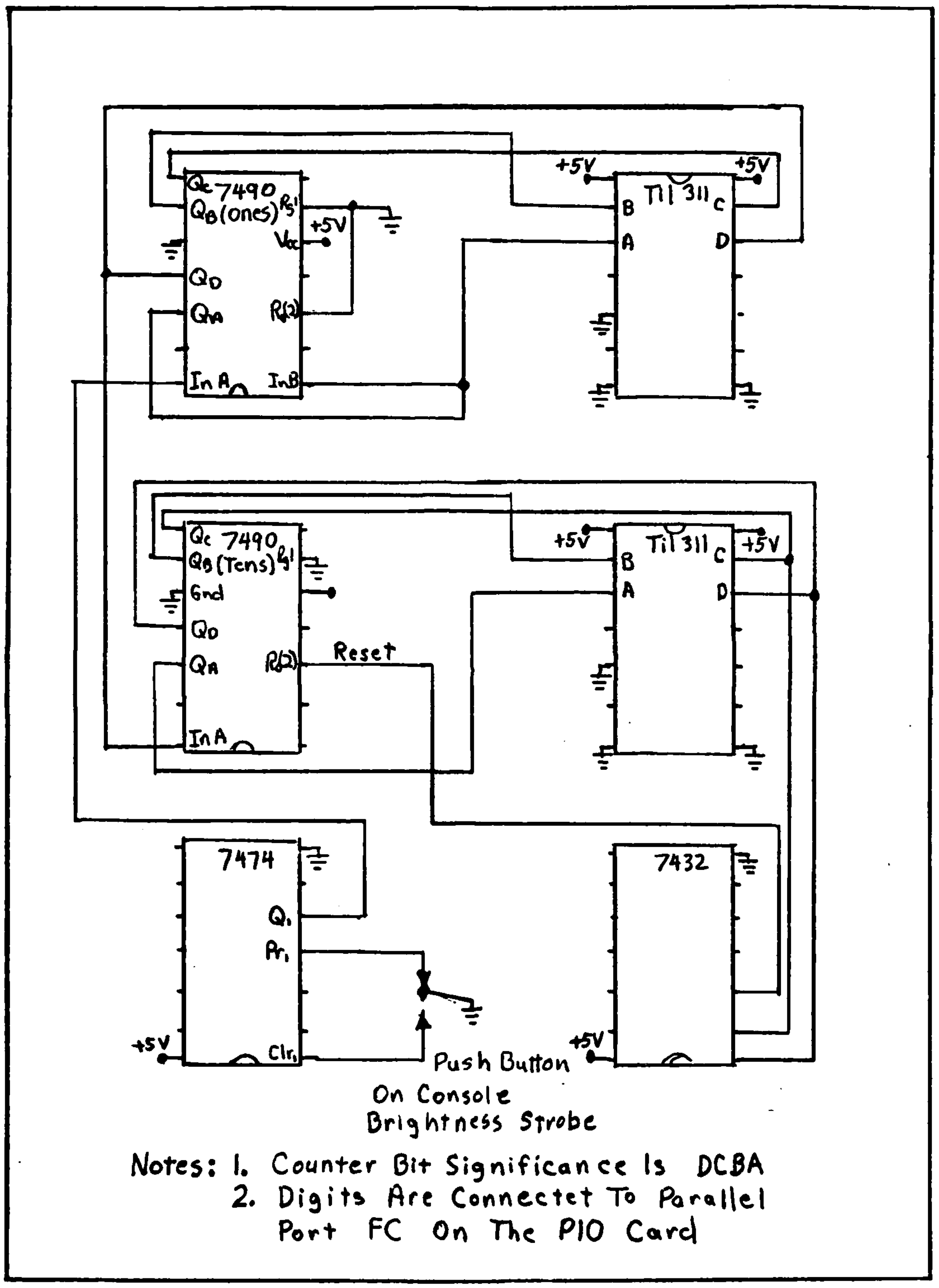
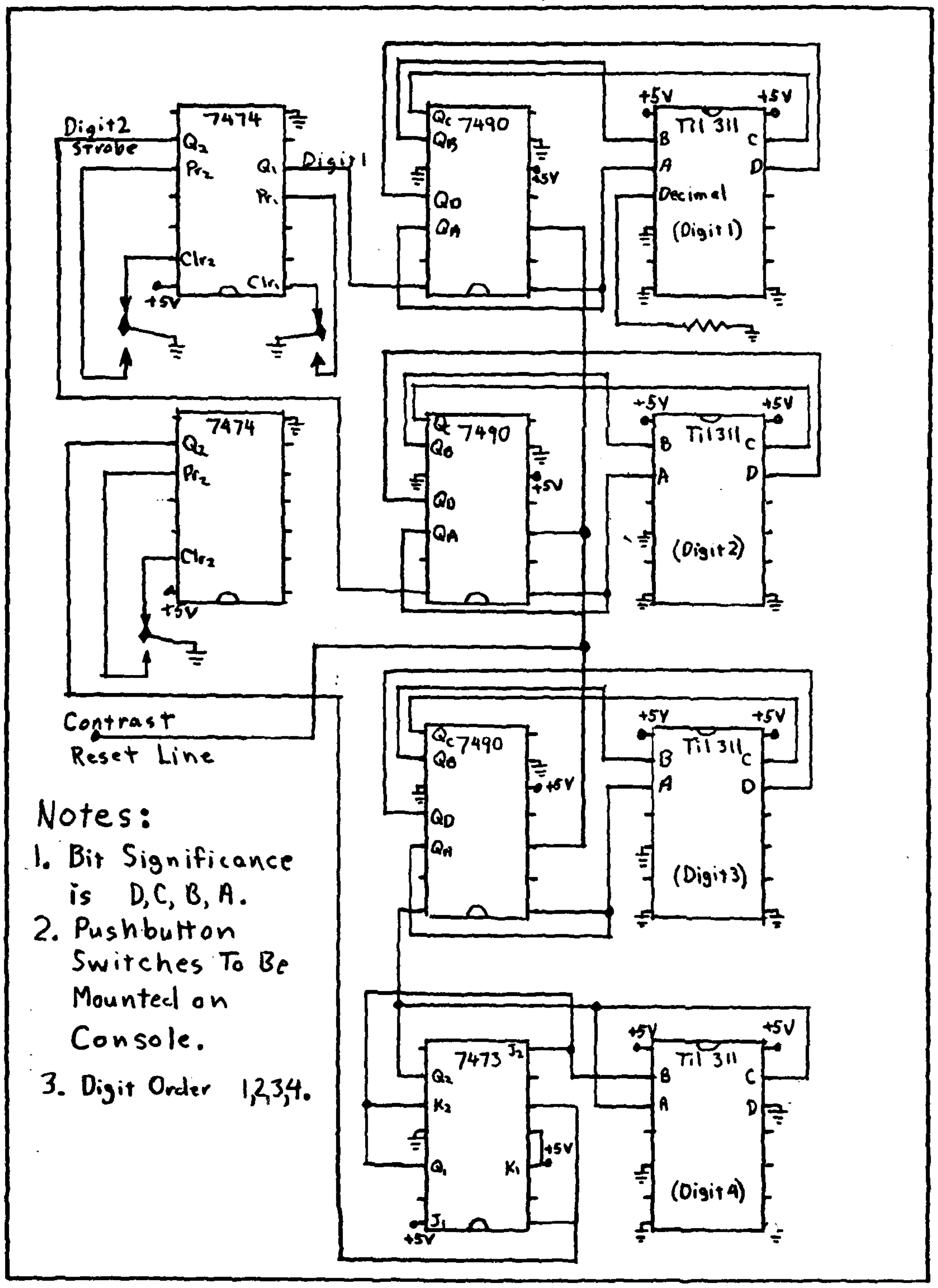


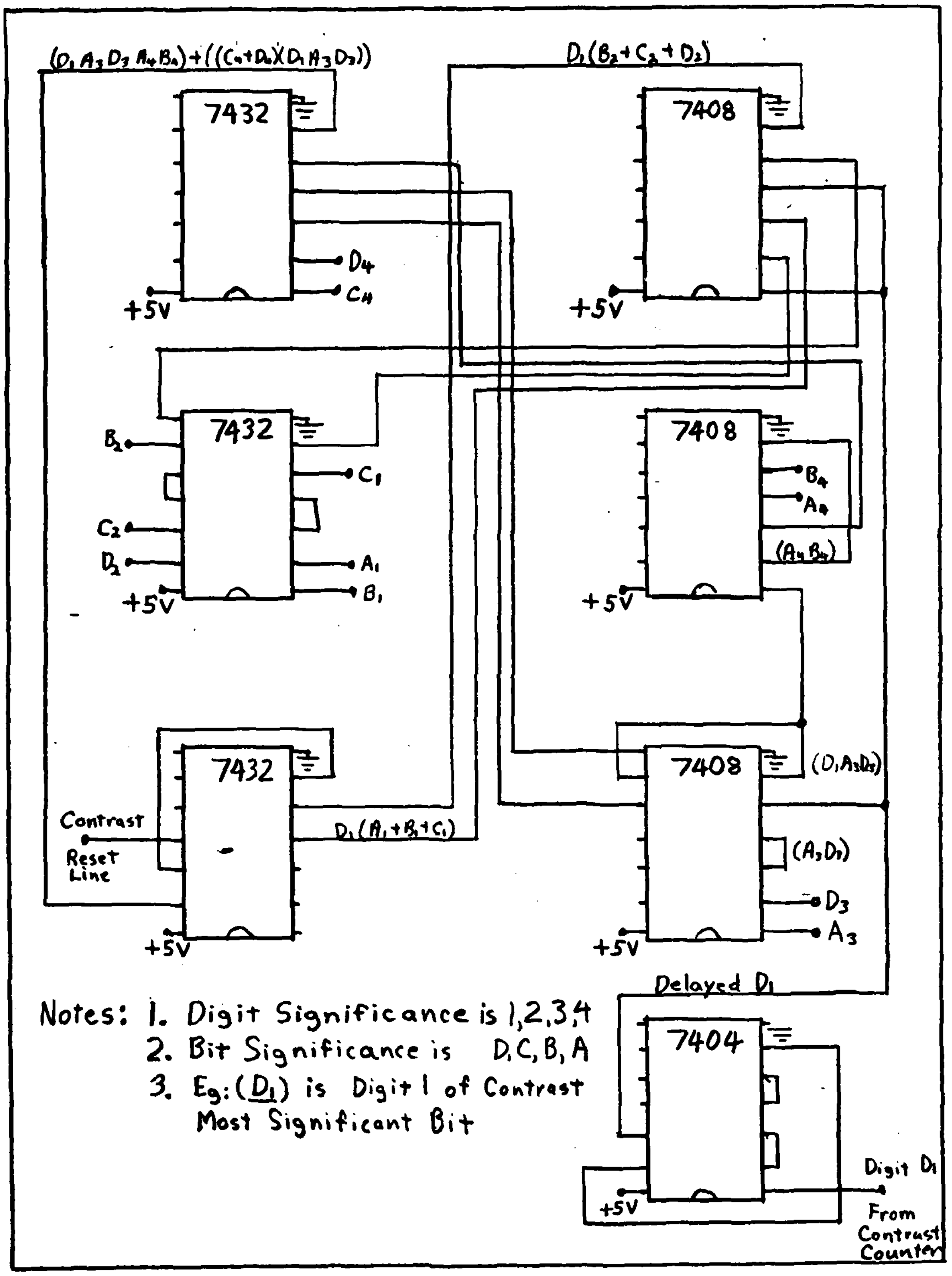
Figure 13. Brightness LED Input Circuit With Reset



Notes:

1. Bit Significance is D, C, B, A.
2. Pushbutton Switches To Be Mounted on Console.
3. Digit Order 1,2,3,4.

Figure 14. Contrast LED Input Circuit



Notes: 1. Digit Significance is 1,2,3,4
2. Bit Significance is D,C,B,A
3. Eg: ($\underline{D_1}$) is Digit 1 of Contrast Most Significant Bit

Figure 15. Contrast Reset Circuit

APPENDIX D

MDX Access Card Drawing

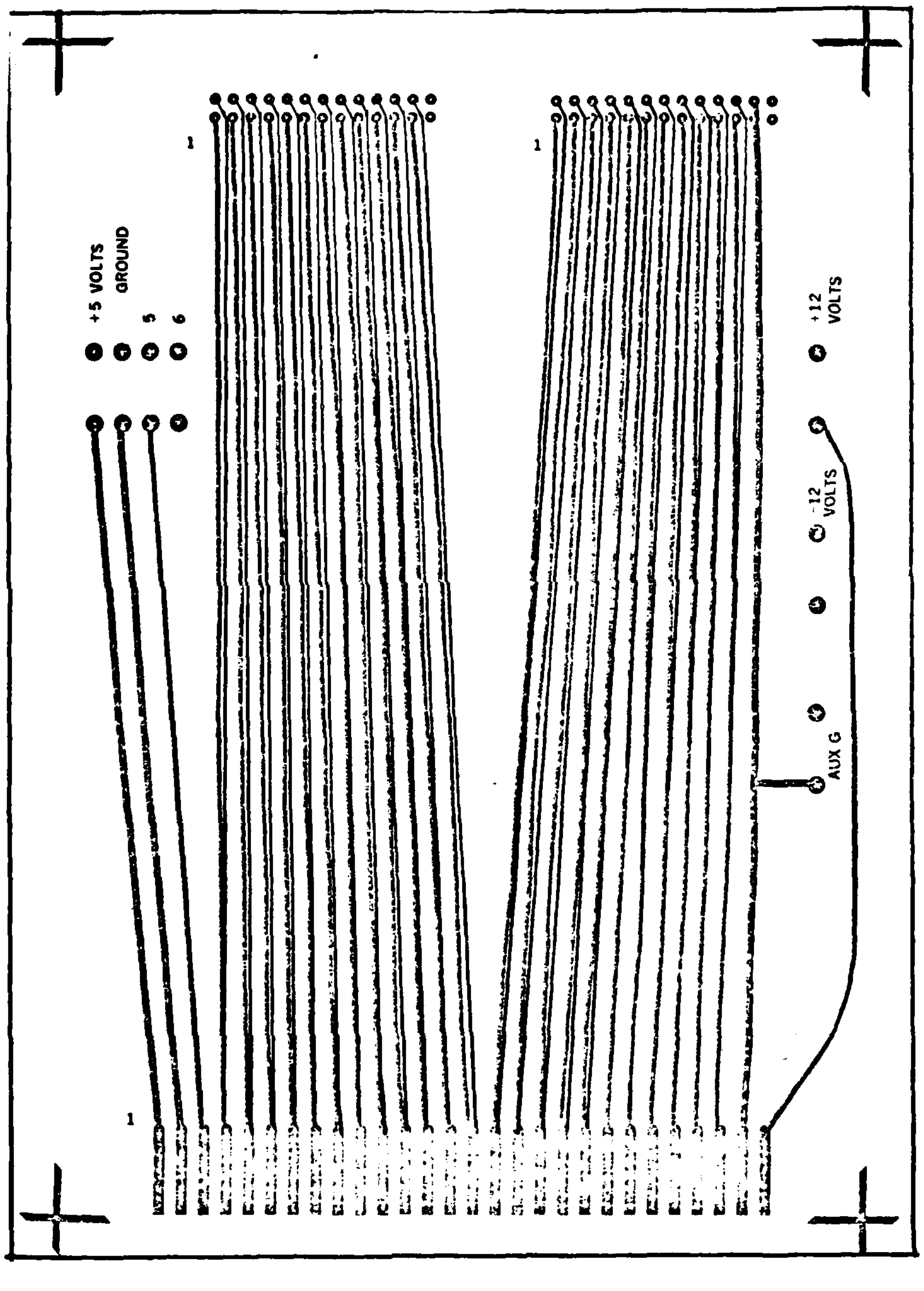


Figure 16. MDX Access Card Drawing

APPENDIX E

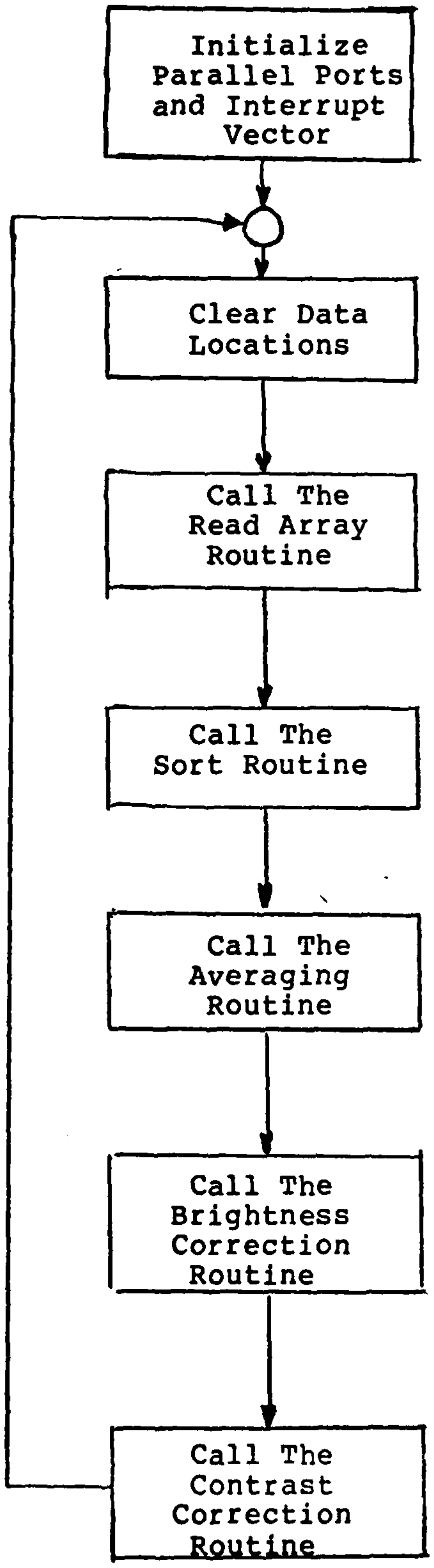
Assembly Language Program

This Appendix includes a flowchart of the basic block structure of the program. Data element definitions are furnished in the comments before each subroutine. The definitions are included there because several data elements are used more than once, with perhaps a different meaning in some of the routines. Most of the data elemente retain the same meaning throughout the program.

The program is well commented to let the reader what each routine and block of data are doing.

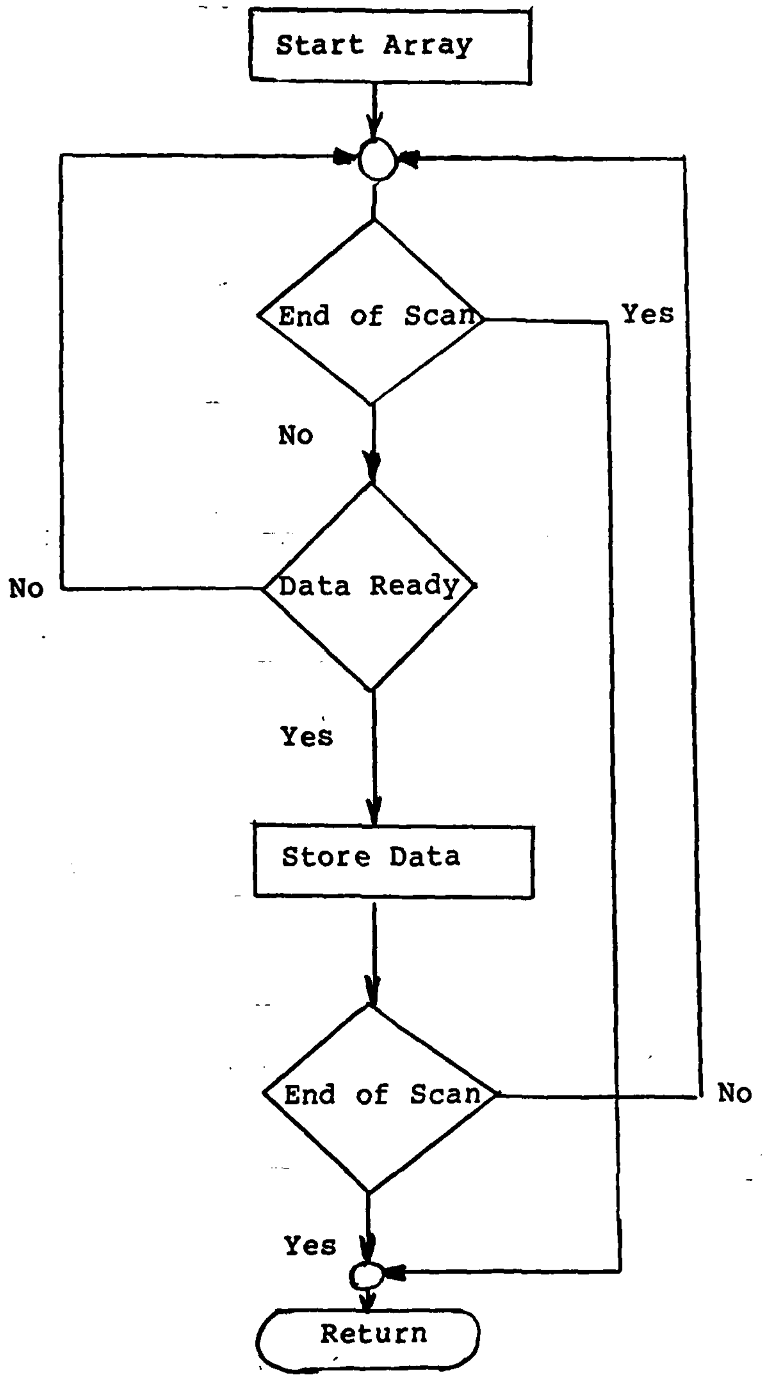
(Program Flowchart)

Main Driver


Initialize Parallel Ports and Interrupt Vector
Clear Data Locations
Call The Read Array Routine
Call The Sort Routine
Call The Averaging Routine
Call The Brightness Correction Routine
Call The Contrast Correction Routine

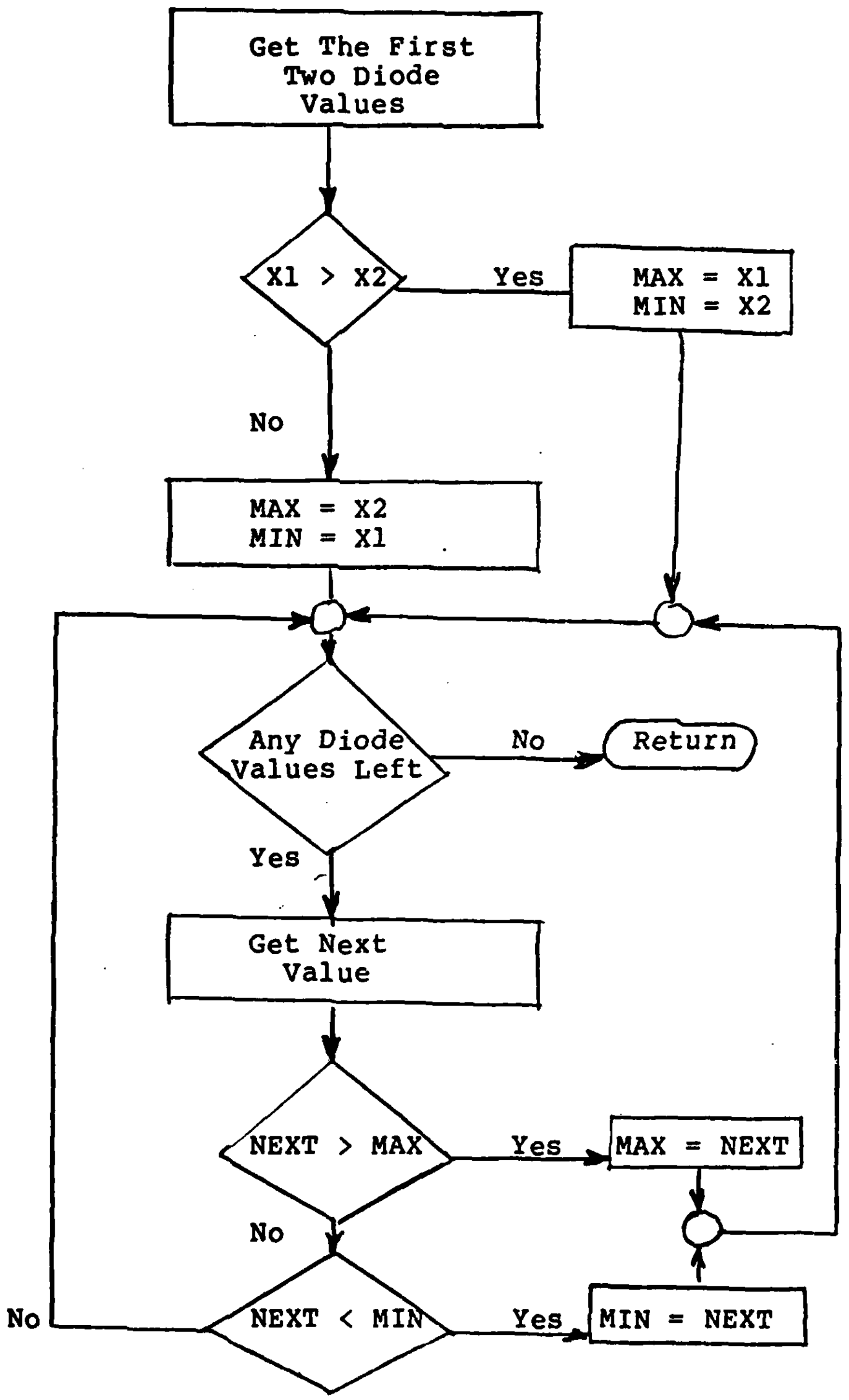
(Program Flowchart)

Read Array Routine



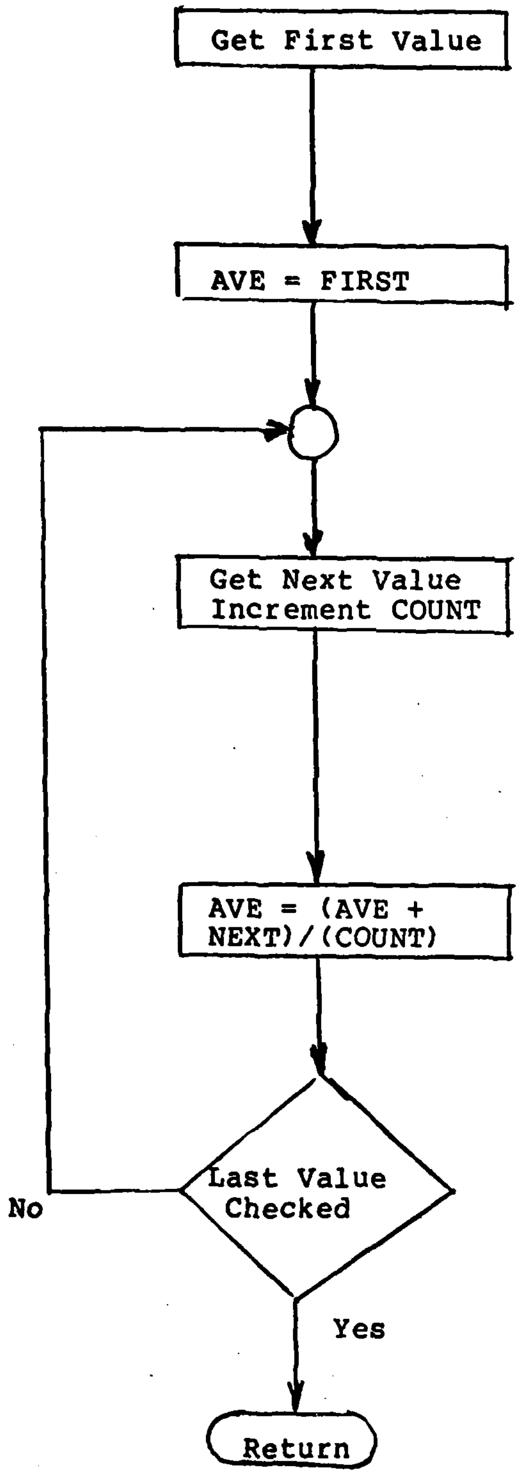
(Program Flowchart)

Sort Routine



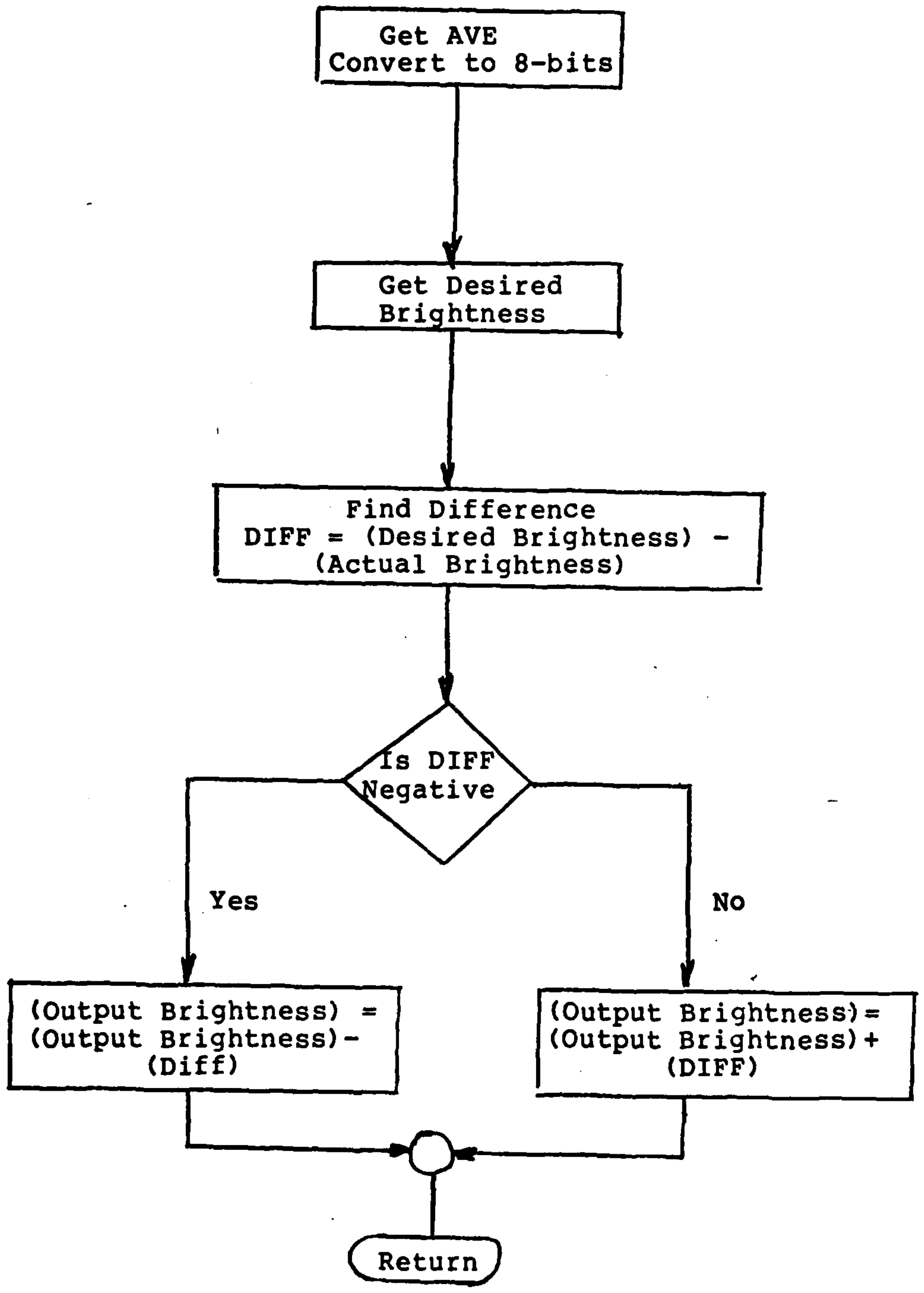
(Program Flowchart)

Average Routine



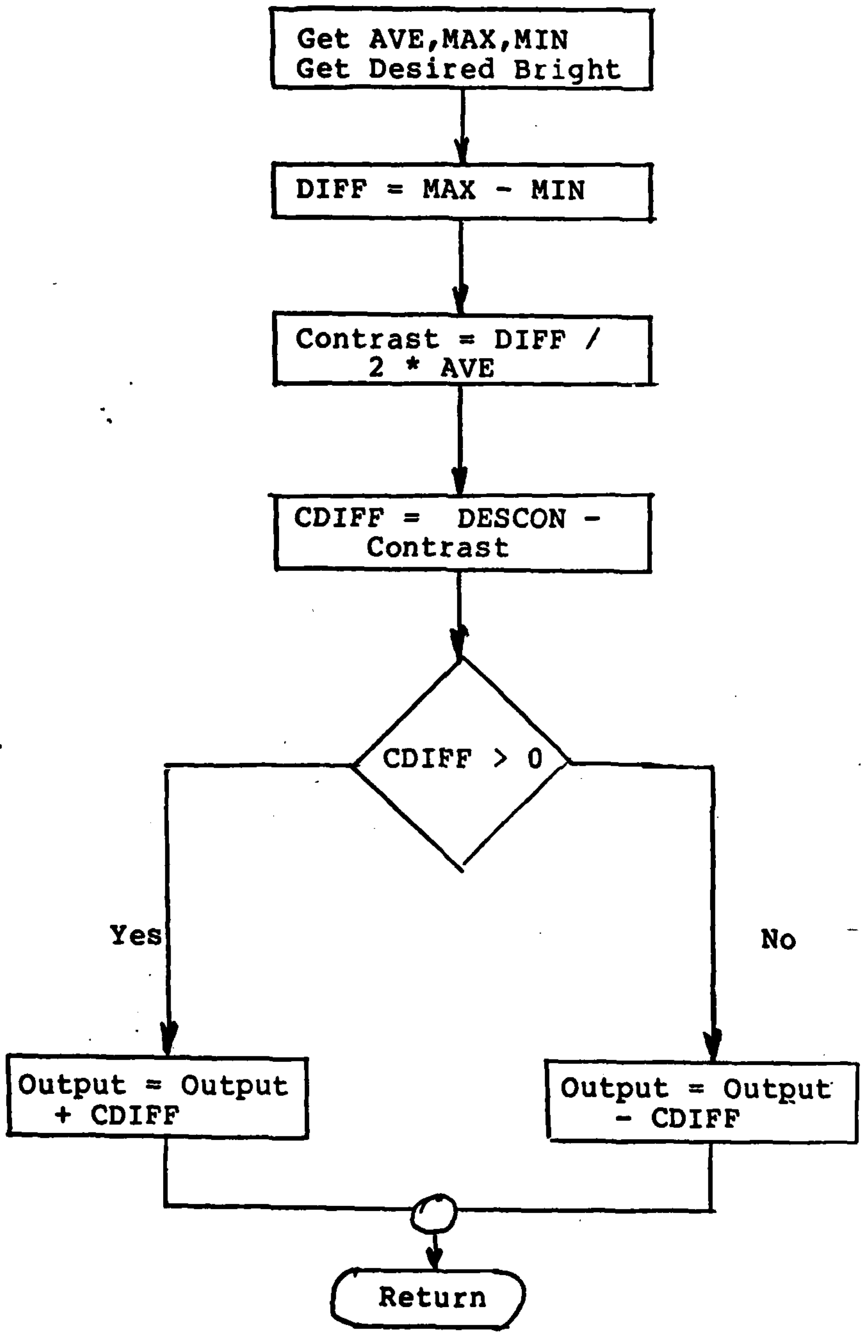
(Program Flowchart)

Brightness Correction


Get AVE
Convert to 8-bits
Get Desired Brightness
Find Difference
DIFF = (Desired Brightness) - (Actual Brightness)
Is DIFF Negative
Yes
No
(Output Brightness) = (Output Brightness) - (Diff)
(Output Brightness) = (Output Brightness) + (DIFF)
Return

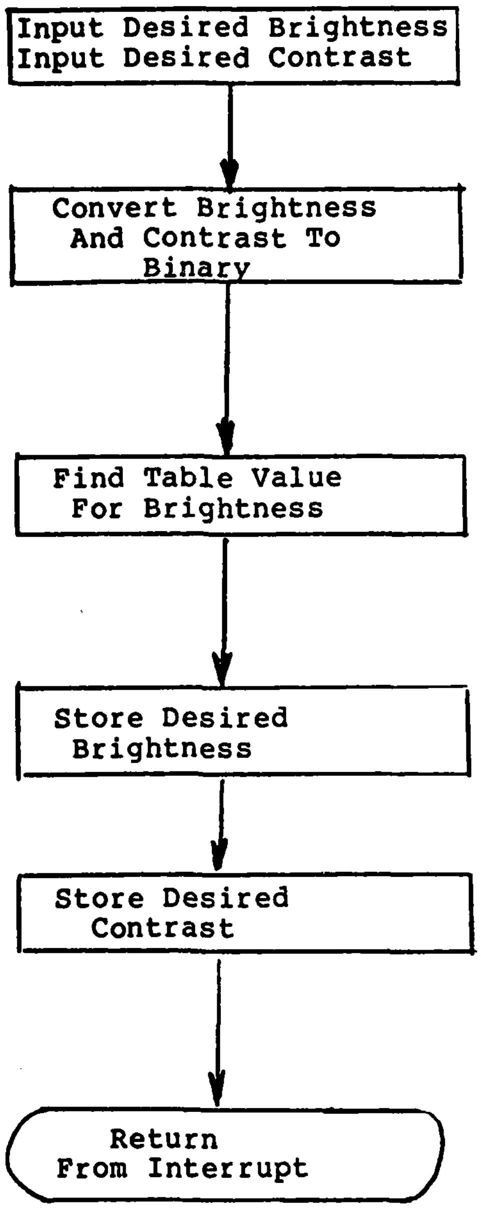
(Program Flowchart)

Contrast Correction



(Program Flowchart)

Interrupt Routine



```
\.Z80
;                Assembly Language Program For
;                     Optronix Controller


ASEG
;         Data Element Locations
ORG 0100H

RANDOM          EQU             0F850H

CDIFF           EQU             0F800H
DIFF            EQU             0F802H
MARK            EQU             0F806H
FIRST           EQU             0F808H
MAX             EQU             0F80AH
MIN             EQU             0F80CH
HOLD            EQU             0F80EH
DIVID           EQU             0F810H
DIVISR          EQU             0F812H
AVE             EQU             0F814H
QUO             EQU             0F816H
DIVIS           EQU             0F818H
REMAIN          EQU             0F81AH
BRITE           EQU             0F81CH
DESIRB          EQU             0F81EH
FIR             EQU             0F81FH
ACTCON          EQU             0F820H
DESCON          EQU             0F822H
SEC             EQU             0F824H
THIRD           EQU             0F825H
FOUR            EQU             0F826H
BRIG            EQU             0F827H
ONES            EQU             0F82AH
FLAG            EQU             0F82BH
COU1            EQU             0F84CH
COUNT           EQU             0F84EH
```

```
;****************************************************************
;                          Main Program
;****************************************************************

START:  DI                ; Disable the interrupts

;               Zero all the flags


        OUT (00H),A       ; RESET THE FLAGS
        OUT (01H),A
        OUT (02H),A


;               Start the initialization of ports

        LD A,04FH         ; Load reg A with input word 4F
;         Initialize the ports with the input word 4F
;                 4F is the output word.
        OUT (0F9H),A      ; Initialize port F8 input
        OUT (0FBH),A      ; Initialize port FA input
        out (0FDH),A      ; Initialize port FC input
        OUT (0FFH),A      ; Initialize port FE input

;               Set the I register with the high byte
;                 of the interrupt vector

        ld a,06H          ; High byte for int vector
        ld I,A            ; Set the I register

;    Set the low byte for port F9H interrupt vector

        ld a,00H          ; Low byte for int vector
        out (0F9H),a      ; Set interrupt enable word

;               Enable the interrupt for port F9H

        ld a,087H         ; Port interrupt enable word
        out (0F9H),a      ; Enable the interrupts

;               Initializt contrast to 00H

        ld h,000H
        ld l,000H
        ld (cdiff),hl
        ld (descon),hl    ; Initialize contrast

;               Initialize brightness to 06H

        ld a,06H
        ld (brite),a
        ld (desirb),a     ; Initialize brightness
```

```
;**************************************************************
;                    Repeating Main Driver
;**************************************************************

INIT:   EI                  ; Enable the interrupt
        LD BC,COU1          ; Put the random address in
                            ;   the C&D registers
        LD E,080H           ; Number of locations zeroed is 80H
        LD D,00             ; Zero the D register

;    ****************************************************
;                    Zero Memory Location Block

REZERO: LD A,00             ; Zero the A register
        LD (BC),A           ; Start zeroing the addresses
        INC BC              ; Increment the B&C registers
        INC D               ; Increment the number of values
        LD A,E              ; Check if enough are zeroed
        CP D                ; yet
        JP NZ,REZERO        ; If not zero another


        DI                  ; Turn off the interrupt
C1:     CALL RARRAY         ; Call read array routine
        EI                  ; enable the interrupt
C2:     CALL SORT           ; Call sorting routine
C3:     CALL AVERG          ; Call averaging routine
        DI                  ; Disable the interrupt
C4:     CALL BRIGHT         ; Call bright correction
        NOP                 ; routine
C5:     CALL CONTR          ; Call contrast correction
        NOP                 ; routine
END:    JP INIT             ; Repeat the program
        HALT                ; End of program

;************************************************************
;               RARRAY Routine
;       This subroutine will read in the values
;         read by the Reticon Card and store them
;         sequentially without interruption.
;                   (RARRAY routine)
;************************************************************

;          ****   Variables Used   ****
; Random: Location F850H, the starting location
;         of array value storage locations
; Count:  Holds the number of array values stored

RARRAY: out (03D),a         ; Start the reticon card read
                            ;   and the clock pulse
        IN A,(0FCH)         ; CHECK if scan started
        bit 6,a
        jp z,rarray         ; Restart if necessary
        LD HL,RANDOM        ; Get address of array
        LD A,00H          ; Zero the A register
        ld (count),a      ; Zero the count

;       *** Check if the value has been read ***

REONE:  in a,(0FCH)       ; Input the status
        bit 1,a           ; Check for end of scan
        jp nz,r2
        bit 0,a           ; Check if value is ready
        jp z,reone        ; If not check again
        bit 1,a           ; Check for end of scan again
        jp nz,r2
        out (02H),a       ; Value is ready reset status
        ld a,(count)      ; Increment the count
        inc a
        ld (count),a      ; Store new count

;  ** Insure that 25 micro seconds (about 100 cycles)
;       have passed to allow the A/D converter enough
;       time to finish it's conversion **

        out (02H),a
        out (02H),a
        out (02H),a
        out (02H),a

; * Adds 48 clock cycles before inputting the value *

        in a,(09D)        ; Input port 9
        cpl               ; Complement the input number
        ld e,a            ; Put number in E register
        in a,(08D)        ; Input port 10
        cpl               ; Complement
        ld d,a            ; Put in D register
        ld (hl),e         ; Store the number
        inc hl            ;
        ld (hl),d         ;
        inc hl            ;

; ** Check if all values are read  **

        in a,(0FCH)       ; Input the status port
        bit 1,a           ; Check if all values read
        jp z,reone        ; If scan not complete get more
R2:     out (01H),a       ; Scan complete, reset status
                          ;   and stop clock count
        ret               ; Return to program
; ** ( End of READ THE ARRAY routine ) **
```

```
;*************************************************************
;                  (SORT subroutine)
;        Subroutine to find max and min values
;*************************************************************

;    ****     Variables Used:    ****

; Random:  Starting location of stored values
; Count:  Number of values stored
; Mark:   Holds location of next value to be read from
;         memory
; First:  Holds the first value read from memory
; Coul:   Counts the number of values read from memory
;         to compare against Count
; Max:    Holds maximum value of those read
; Min:    Holds minimum valuue of those read
; Hold:   Holds HL register contents to retain after
;         subtraction
;

SORT:   LD HL,RANDOM      ; LOAD Array address
        ld (mark),hl      ; Set MARK = ARRAY
        ld e,(hl)         ; Get first value
        inc hl
        ld d,(hl)
        inc hl
        ld (mark),hl      ; Store new value for MARK
        ld (first),de     ; Store the first value
        ld a,02           ; Set the count to two
        ld (coul),a       ; Set  COUNT
        ld hl,(mark)      ; Get the second value
        ld e,(HL)
        inc hl
        ld d,(hl)
        inc hl
        ld (mark),hl      ; Store new value for MARK
        ld hl,(first)     ; Get the first value
        scf
        ccf
        ld (hold),hl
        sbc hl,de         ; Find the larger number
        ld hl,(hold)
        jp m,small        ; If the first value is larger
                          ; then move it to D&E reg's
        ex de,HL          ; Exchange the values
SMALL:  ld (max),de       ; Store as max value
        ld (min),hl       ; Store as min value
GET:    ld hl,(mark)      ; Get next value address
        ld e,(hl)         ; Get the next value
        inc hl
        ld d,(hl)
        inc hl
        ld (mark),hl      ; Store new address
```

```
;           Check the next value against the
;                curent maximum value

        ld hl,(max)       ; Get the current maximum
        scf
        ccf
        ld (hold),hl
        sbc hl,de         ; Find the smaller value
        ld hl,(hold)
        jp m,sml          ; Store D&E as MAX

;                Check against current minimum value

SM:     ld hl,(min)       ; Check the new value against
        scf
        ccf
        ld (hold),hl
        sbc hl,de         ; Find larger value
        ld hl,(hold)
        jp m,sm3
        ld (min),de
        jp sm3            ; No change get next value

SM1:    ld (max),de       ; Store the largest in MAX
SM3:    ld a,(coul)       ; Increment the count
        inc a
        ld (coul),a
        ld a,(count)
        ld b,a
        ld a,(COUl)       ; Check if all values checked
        scf
        ccf
        sub b
        jp m,get          ; If not get the next
Rl:     ret               ; Return if counts are equal

;       ** ( End of the SORT subroutine ) **
```

```
;***************************************************************
;                    AVERG Routine
;                (Averaging subroutine)
;            This is the subroutine to find the
;               average value  (or brightness)
;**************************************************************

;          **** Variables Used   ****

; Count:  Number of stored values from array
; Random: Starting location of the stored values
; Mark:   Holds address of next value to be read
; First:  Holds first value read
; AVE:    Holds the average value of the values read
;         from memory to current value
; Diff:   Holds difference between current average
;         value (AVE) and the next value read
;         from memory
;
;          ****  Routines Used    ****

; Sign:   Sets bit 0 of location flag to note negative
;          number
; Divide: Divides the DE registers by the BC registers
;         returning the answer in the DE registers
; Error:  Keeps AVE from going negative


AVERG:  ld c,01H
        ld d,00H
        ld (coul),bc     ; Set count to 01
        ld hl,RANDOM     ; Set H&L reg's to first
        ld (mark),hl     ; Set marker value
        ld e,(HL)        ; Get first number LSB's
        inc hl
        LD D,(HL)        ; Get MSB's
        inc hl
        ld (first),de    ; Store first number (FIRST)
        ld (ave),de      ; One value is first average
        exx
        res 4,d          ; Reset the fraction flag
        exx
        ld (mark),hl     ; Update marker
RET2:   LD A,00H         ; Zero the flag
        ld (flag),a
        ld hl,(mark)     ; Set H&L reg's
        ld e,(hl)        ; Get next number LSB's
        inc hl
        ld d,(hl)        ; Get MSB's
        inc hl
        ld (mark),hl     ; Reset marker
        ld bc,(coul)
        inc bc           ; Update count
        ld (coul),bc
        ld hl,(AVE)      ; Get the first number
```

```
         scf
         ccf
         SBC HL,DE        ; Find the difference
         call m,sign      ; Set the sign flag
                          ;  if necessary
         ld (diFF),hl     ; Set dividend
         ld de,(diff)
         scf
         ccf
         rl e             ; Set up to divide number
         rl d
         rl e
         rl d
         rl e
         rl d
         rl e
         rl d
         ld bc,(coul)     ; Set divisor
         ld (divis),bc
         call divide      ; Call division routine
         ld a,(flag)      ; Get the flag
         bit 0,a          ; Check if minus flag set
         LD HL,(AVE)      ; Get the average value
         jp Nz,add
         scf
         ccf
         sbc hl,de
         call m,(error)   ; If minus set AVE = zero
         jp al
ADD:     add hl,de
Al:      ld (ave),hl      ; Store new average value
         ld bc,(COUNT)    ; Get number of values
         ld hl,(coul)     ; Get current count
         sbc hl,bc        ; Check if finished
         jp m,ret2        ; Not finished get another
                          ;    value
RETl:    ret              ; END averaging routine

;        ( End of averaging subroutine)
```

```
;************************************************************
;                  Set HL registers to zero
ERROR:  ld h,00H
        ld l,00H
        ret

;***********************************************************
;                      SIGN Routine
;       This routine is internal to AVERG and takes
;        the two's complement of a negative number
;___________________________________________________________

SIGN:   set 0,a          ; Set the flag
        ld (flag),a
        scf
        ccf
        ld a,l           ; Take the two's complement
        cpl
        ld l,a
        ld a,h
        cpl
        ld h,a
        inc hl
        ret

;************************************************************
;                          BRIGHT Routine
;                  Routine to make the brightness correction
;************************************************************

;  ****         Variables Used        ****

; DESIRB: Holds the desired brightness value
; BRITE:  Holds the last value of brightness output
;         to the generator

;         ****  Routines Used   ****

; Check:  Sets a flag bit for negative number sign
; Both:   Rotates B and H registers one bit right
; Left:   Rotates H register one bit left

BRIGHT: ld HL,(ave)      ; Get the average value
        RL L
        rl h             ; Change to 8-bit value
        rl l
        rl h
        rl l
        rl h
        rl l
        rl h
        ld a,(desirb)    ; Get desired brightness
        ld b,a           ; Move to B register
        res 6,d          ; Zero the flag
        bit 7,b          ; Check sign bit of B
        call Nz,check    ; Set the flag if negative
        bit 7,h          ; Check sign of H
        call nz,check    ; Set flag if negative
        bit 6,d          ; Check the flag
        call nz,both     ; Flag set move both over
        ld a,b           ; Move desired to A register
        sub h            ; Find offset
        jp m,subl        ; Change routine if negative
        ld h,a           ; Put difference in H reg
        bit 6,d          ; Check flag again
        call nz,left     ; Move left if set
        res 6,d          ; Zero flag again
        ld a,(desirb)    ; Get current brightness
        ld b,a           ; Move to B register
        bit 7,b
        call nz,check    ; Set flag if negative
        bit 7,h          ; Check sign
        call nz,check
        bit 6,d          ; Check the flag
        call nz,both     ; Move both right if set
        ld a,b           ; Move to A register
        add a,h          ; Make brightness correction
        ld h,a           ; Put in H register
        bit 6,d          ; Check the flag again
        call Nz,left     ; Move left if set
        ld a,h           ; Move back to A
        jp Bl            ; Skip subl
SUBl:   cpl              ; Complement the value
        inc a            ; add one
        ld h,a
        bit 6,d          ; Check the flag
        call nz,left     ; Move left if set
        res 6,d          ; Zero the flag
        ld a,(brite)     ; Get the current output
        ld b,a
        bit 7,b          ; Check the sign bit
        call nz,check    ; Set the flag if negative
        bit 6,d          ; Check the flag
        call nz,both     ; Move right if set
        ld a,b
        sub h            ; Subtrtact the difference
        bit 7,a          ; Check the sign
        jp z,b2          ; Output if positive
        ld a,000H        ; Zero A if negative
        jp bl
B2:     ld h,a
        bit 6,d
        call nz,left
        ld a,h
Bl:     ld (brite),a     ; Store the new value
        out (004H),a     ; Output the value
        ret              ; Return from brightness rooutine
```

```
;*******************************************************
;               CHECK sets a flag bit
;_______________________________________________________
CHECK:  set 6,d                           ; Set the flag
        ret

;*******************************************************
;       BOTH rotataes the A and H registers right
;_______________________________________________________
BOTH:   scf             ; Rotate A and H right one
        ccf
        rr B
        scf
        ccf
        rr h
        ret

;*******************************************************
;               LEFT rotates H register left
;_______________________________________________________
LEFT:   scf             ; Rotate H left one
        ccf
        rl h
        ret

;*****************************************************
;                 CONTR Routine
;  Routine to make the required contrast correction
;**********************************************************

;         ****   Variables Used    ****

; Max:    Largest value stored
; Min:    Smallest value stored
; Diff:   Difference between MAX and MIN
; CDIFF:  Difference between actual contrast and
;         desired contrast
; Descon: Desired contrast value (binary)
; Actcon: Actual contrast value calculated by routine
; Hold:   Holds HL register value temporarally


CONTR:  ld hl,(max)     ; Get max value
        ld bc,(min)     ; Get minimum value
        scf
        ccf
        sbc hl,bc       ; Find the difterence
        ld (diff),hl    ; Store the difference
        ld bc,(ave)     ; Get the average value
        scf             ; Multiply by 2
        ccf
        rl c
        rl b
        ld (divis),bc   ; Store new value
        ld de,(diff)
        exx
        set 4,d         ; Set the flag for fraction
        exx
;       The average (AVE) now equals twice brightness

;       Divide difference by (B max + B min)
;       The answer is the contrast value of screen


        call divide
        ld a,d          ; Zero the first 4 bits
        and 0FH
        ld d,a
        ld (ACTCON),de  ; Store contrast
        ld hl,(descon)  ; Get desired output
        sbc hl,de       ; Find offset
        ld de,(descon)  ; Get last output
        add hl,de       ; Make correction
        ld (cdiff),hl   ; Store new value
        ld a,l          ; Output the contrast difference
        cpl
        out (05D),a     ; Output LSB's
        ld a,h
        cpl
        out (06D),a     ; Output MSB's
        ret             ; Return

;**********************************************************
;              DIVIDE makes a division by shifting
;       the registers to the left and setting
;       the answer register one bit at a time
;__________________________________________________________


;     _    ****      Routines Used     ****

; Frac:   Does first subtraction for DIVIDE
; Fract:  Rotates DE registers into HL registers
; Chsign: Checks sign bit of HL an BC registers,
;         seting a flag if they are negative
; Digres: Starts process of resetting a bit of the
;          DE registers during DIVIDE
; Digset: Starts process of setting a bit of the
;          DE registers during DIVIDE
DIVIDE: ld h,00H
        ld l,00H
        exx
        bit 4,d         ; Check if number is fraction
        exx
        jp nz,frac
        call fract      ; Move the first bit into H&L
        call chsign     ; Check the sign bits
        ld (hold),hl    ; Hold the value
        sbc hl,bc       ; Start division
        jp m,res3d      ; Dividend smaller
        exx
        set 3,d         ; Dividend larger set MSB
        call digres
        jp set2d        ; Determine next smaller bit
RES3D:  ld hl,(hold)
        exx
        res 3,d         ; Zero the MSB
        call digres
SET2D:  call digset     ; Determine bit 10
        jp m,res2d
        exx
        set 2,d
        call digres
        jp set1d
RES2D:  ld hl,(hold)
        exx
        res 2,d
        call digres
SET1D:  call digset     ; Determine bit 9
        jp m,res1d
        exx
        set 1,d
        call digres
        jp set0d
RES1D:  ld hl,(hold)
        exx
        res 1,d
        call digres
SET0D:  call digset     ; Determine bit 8
        jp m,res0d
        exx
        set 0,d
        call digres
        jp set7e
RES0D:  ld hl,(hold)
        exx
        res 0,d
        call digres
SET7E:  call digset     ; Determine bit 8
        jp m,res7e
        exx
        set 7,e
        call digres
        jp set6e
RES7E:  ld hl,(hold)
        exx
        res 7,e
        call digres
SET6E:  call digset     ; Determine bit 7
        jp m,res6e
        exx
        set 6,e
        call digres
        jp set5e
RES6E:  ld hl,(hold)
```

```
        exx
        res 6,e
        call digres
SET5E:  call digset     ; Determine bit 5
        jp m,res5e
        exx
        set 5,e
        call digres
        jp set4e
RES5E:  ld hl,(hold)
        exx
        res 5,e
        call digres
SET4E:  call digset     ; Determine bit 4
        jp m,res4e
        exx
        set 4,e
        call digres
        jp set3e
RES4E:  ld hl,(hold)
        exx
        res 4,e
        call digres
SET3E:  call digset     ; Determine bit 3
        jp m,res3e
        exx
        set 3,e
        call digres
        jp set2e
RES3E:  ld hl,(hold)
        exx
        res 3,e
        call digres
SET2E:  call digset     ; Determine bit 2
        jp m,res2e
        exx
        set 2,e
        call digres
        jp set1e
RES2E:  ld hl,(hold)
        exx
        res 2,e
        call digres
SET1E:  call digset     ; Determine bit 1
        jp m,res1e
        exx
        set 1,e
        call digres
        jp set0e
RES1E:  ld hl,(hold)
        exx
        res 1,e
        call digres
SET0E:  call digset     ; Determine bit 0
        jp m,res0e
        exx
        set 0,e
        call digres
        exx
        jp setde
RESOE:  ld hl,(hold)
        exx
        res 0,e
SETDE:  ret                 ; End of division routine

;**********************************************************
;       FRAC sets regesters for a fractional division
;         call to be done by DIVIDE
;_________________________________________________________
FRAC:   ex de,hl            ; Put number in H&L regs
        ld (hold),hl        ;    and start the division
        sbc hl,bc           ;    process.
        jp m,res3d
        exx
        set 3,d
        exx
        jp set2d

;*********************************************************
;               DIGSET Sets up registers to continue to
;                 divide a number or fraction
;________________________________________________________
DIGSET: call fract          ; Set up for fraction
        call chsign         ; Check sign bits
        ld (hold),hl
        sbc hl,bc
        ret

;********************************************************
;               part of division
;               reset of answer bit here
;_______________________________________________________
DIGRES: bit 6,d
        exx
        call nz,movel
        ld bc,(divis)
        ret

;*******************************************************
;               Checks sign of H and B registers
;_______________________________________________________
CHSIGN: exx
        res 6,d
        exx
        bit 7,h             ; Check sign bit
        exx
        call nz,check       ; Set flag if negative
        exx
        bit 7,b
        exx
        call nz,check
        bit 6,d         ; Check flag
        exx
        call nz,mover   ; Move right if set
        ret

;*****************************************************
;               Moves HL and BC right one bit
;____________________________________________________
MOVER:  scf             ; Move HL and BC numbers right
        ccf
        rr h
        rr l
        scf
        ccf
        rr b
        rr c
        ret


;*****************************************************
;               Moves HL left one bit
;____________________________________________________
MOVEL:  scf             ; Move HL number left
        ccf
        rl l
        rl h
        ret

SET:    scf             ; Set the carry flag
        ret             ; Return

;*****************************************************
;       Rotates HL and DE left for next division step
;____________________________________________________
FRACT:  scf
        ccf
        rl e
        rl d
        rl l
        rl h
        ld (diff),de
        ret
```

```
;***********************************************************
;                         INTADD Routine
;       This is the routine to input the desired values of
;    brightness and contrast.  The routine is an interrupt
;    service routine, which is called by the data strobe
;    signal to port F8, and is input by switch.
;         The values arer input in decimal form and converted
;    to binary form by the routine.
;         Brightness is then changed to a table value.
;**********************************************************

;         ****      Variables Used      ****

; FIR:    Holds desired brightness decimal digits
; SEC:    Holds desired contrast decimal digits
;            (ones and tens)
; Third:  Holds desired contrast decimal digits
;            (hundreds and thousanas)
; Desirb: Binary value of desired brightness
; Descon: Binary value of desired contrast


ORG 0600H
        002H             ; Interrupt vector location
        006H

INTADD: in a,(0F8H)      ; Input the brightness digits
        ld (fir),a       ;    Ones and tens digits
        in a,(0FAH)      ; Input contrast smallest digits
        ld (sec),a       ;  Ones and tens digits
        in a,(0FEH)      ; Input contrast largest digits
        ld (third),a     ;     Store number

;   The input digits are stored now determine the
;    binary values

        ld a,(fir)       ; Get brightness digits
        and 00FH         ; Zero the four MSB's
        ld b,a           ; Put ones in B register
        ld a,(fir)
        call rotat       ; Move the ten's digit right

;   The B register now has the number of tens
;   now add the tens to the ones.

        ld d,000H        ; Zero the D register
        ld e,010D        ; Put 10 in the E register
        call hit         ; Convert to binary

;   The B register now has the correct value so look
;  up the correct table value after making sure the
;  value is 39 or less.

        ld a,b
        cp 039D
        call p,changb    ; If larger get 39 decimal
        ld hl,table      ; Get the table address
        ld e,b           ; Put the brightness table offset
        ld d,000H        ;     in the D&E registers
        scf              ; Zero the carry flag
        ccf
        add hl,de        ; Increment to correct table value
        ld a,(hl)        ; Get the brightness value
        ld (desirb),a    ; Store the desired value of
                         ; brightness in DESIRB

;   Get and store the contrast difference value

        ld a,(sec)       ; Get contrast
        and 00FH         ;     One's and ten's
        ld b,a           ; Save the one's value in B
        ld a,(sec)
        call rotat       ; Get the # of ten's
        ld d,000H        ; Zero the D register
        ld e,010D        ; Put 10 decimal in E
        call hit         ; Convert to binary equivalent

        ld a,b           ; Store the binary value of the
        ld (ones),a      ;   2 digits in ones

        ld a,(third)     ; Get contrast 100's & 1000's
        and 00FH         ; Get # of 100's
        ld b,a           ; Save # of 100's in B
        ld a,(third)     ;
        call rotat       ; Get # of 1000's

        ld d,000H        ; Zero the D register
        ld e,010D        ; Put 10 decimal in E
        call hit         ; Convert to binary equivalent

        ld c,b           ; Store # of 100's in C register
        ld a,(ones)      ; Get the # of ones
        ld l,a
        ld h,000H        ; Put # ones in H&L registers


        ld e,0100D       ; Put 100 decimal in E register
        ld d,000H        ; Zero the D register
        ld b,000H        ; Zero the B register
;**********************************************************
;                 Make the whole number binary

HITl:   add hl,de        ; Add 100 to the ones
        inc b
        ld a,b           ; Load current # of 100's added
        cp c             ; Check if all 100's added
        jp m,hitl        ; Repeat if more 100's needed
```

```
;   See if the number is larger than .8192 decimal

        ld de,08191D    ; Load 8191 in D&E regs
        ex de,hl
        sbc hl,de
        call m,changc   ; If larger change contrast
        ex de,hl
        scf
        ccf
        rr h            ; Rotate H&L regs one bit right
        rr l
        ld (descon),hl  ; Load the desired contrast
                        ;  value in DESCON
        out (00),a      ; Reset interrupt signal
        reti            ; Return from the interrupt

;*********************************************************
;               Maximum allowed is 39 for brightness

CHANGB: ld b,039D       ; Put 39 in B register
        ret

;********************************************************
;               Maximum allowed is .8192 for contrast

CHANGC: ld de,08191D    ; Put 8191 in D&E registers
        ret

;*******************************************************
;       Move the tens bit into the ones bit spot
;         by rotating it right four bits

ROTAT:  and 0F0H        ; Get the tens digit
        scf
        ccf
        rrc a           ; Move tens digit to the right
        rrc a
        rrc a
        rrc a
        ld c,a          ; Put tens in C register
        ret




;********************************************************
;               Convert to a binary number

HIT:    ld a,b          ; Get the ones again
        add a,e         ; Add in the tens digits
        ld b,a          ; Store the new value
        inc d
        ld a,d
        cp c            ; See if ten tens were added
        jp m,hit        ; If not ten add another
        ret


ORG 700
TABLE:  ;The table of values are stored starting at
          ;  location 700H.

END    0100H
```

APPENDIX F

Modulo 15 Counter Design

AD-A115 502   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH   SCHOO--ETC   F/G 20/6
PROGRAMMED CONTROL OF A SINE-WAVE GRATING GENERATOR FOR VISUAL --ETC(U)
DEC 81   A L LAWSON
UNCLASSIFIED   AFIT/GE/EE/81D-33   NL

2 OF 2
AD
A115502

END
DATE
FILMED
7-82
DTIC

Modulo 15 Counter Design

The modulo 15 counter was designed using four J-K Flip-flops. The design procedure used is outlined in Reference 7 and follows that format directly. No external inputs were required to the counter other than the clock pulse which was logically anded with the inverted Reticon blanking pulse. Two outputs were required which were obtaited using the same design procedure. The J and K input equations are shown below.

$$JA = BCD \tag{1}$$

$$KA = BC \tag{2}$$

$$JB = CD \tag{3}$$

$$KB = CD + AC \tag{4}$$

$$JC = D \tag{5}$$

$$KC = D + AB \tag{6}$$

$$JD = C' + A' + B' \tag{7}$$

$$KD = 1 \tag{8}$$

The A flip-flop represented the least significant bit of the counter, B the next least, C the next, and D the most significant bit. Table 1 was used to set up the Karnaugh Maps used to determine the equations for the J and K inputs of the flipflops. Outputs were required on counts 13 and 14 of the counter, and were also determined with Karnaugh Maps with a zero in all but the desired output state to prevent the generation of a pulse at any but the desired count.

The logical equations for the outputs are:

$$\text{Count 13} = ABD \tag{9}$$

$$\text{Count 14} = ABC \tag{10}$$

TABLE 1

State Table used to determine the Karnaugh Maps for the Modulo 15 counter.

Present State				Next State				Inputs To Flipflops							
A	B	C	D	A	B	C	D	JA	KA	JB	KB	JC	KC	JD	KD
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
1	1	1	0	0	0	0	0	X	1	X	1	X	1	0	X

The circuits used to generate these logical equations are shown in Figure 17.

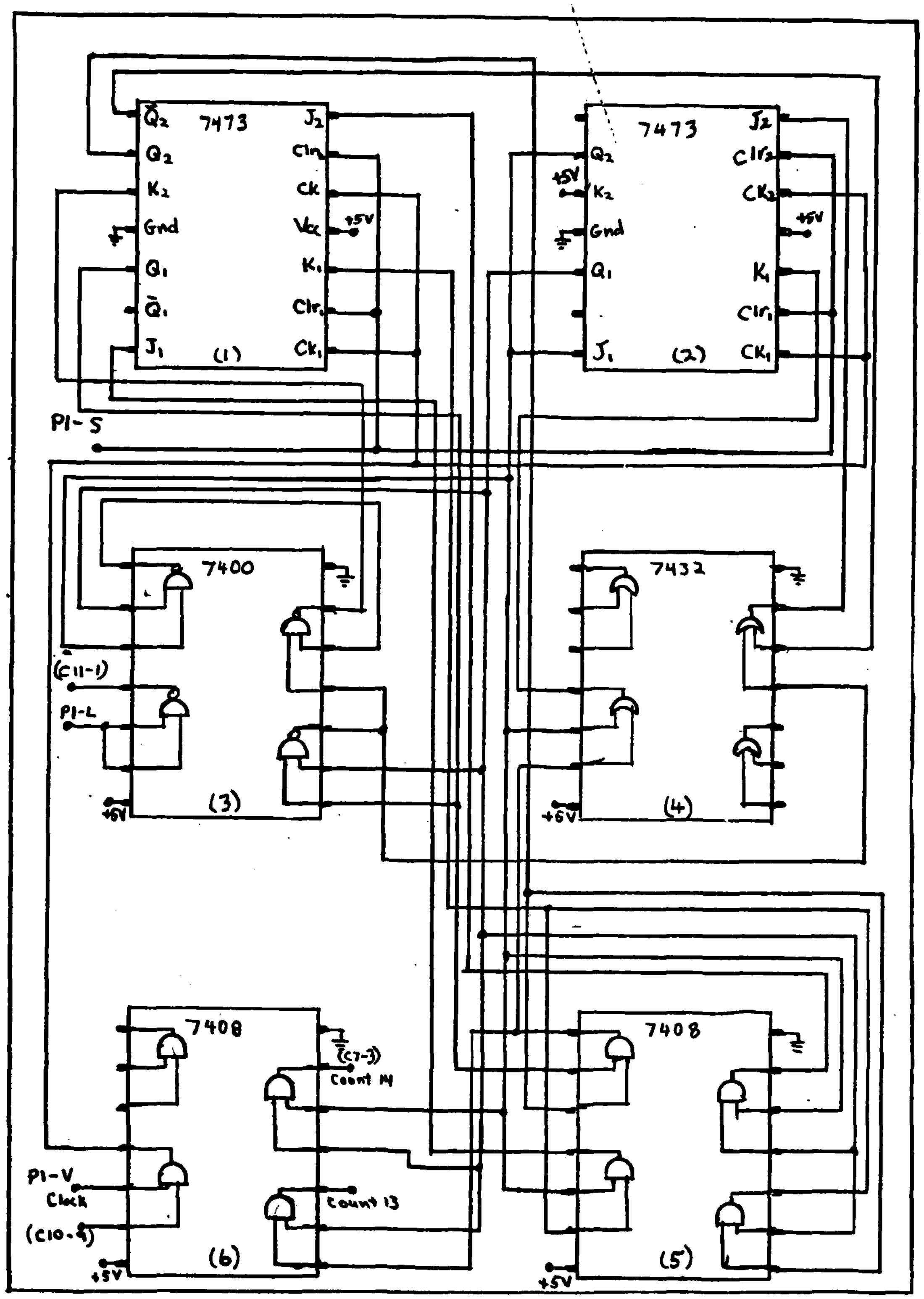


Figure 17. Modulo 15 Counter Circuit

APPENDIX G

WIRE WRAP BOARD DRAWINGS

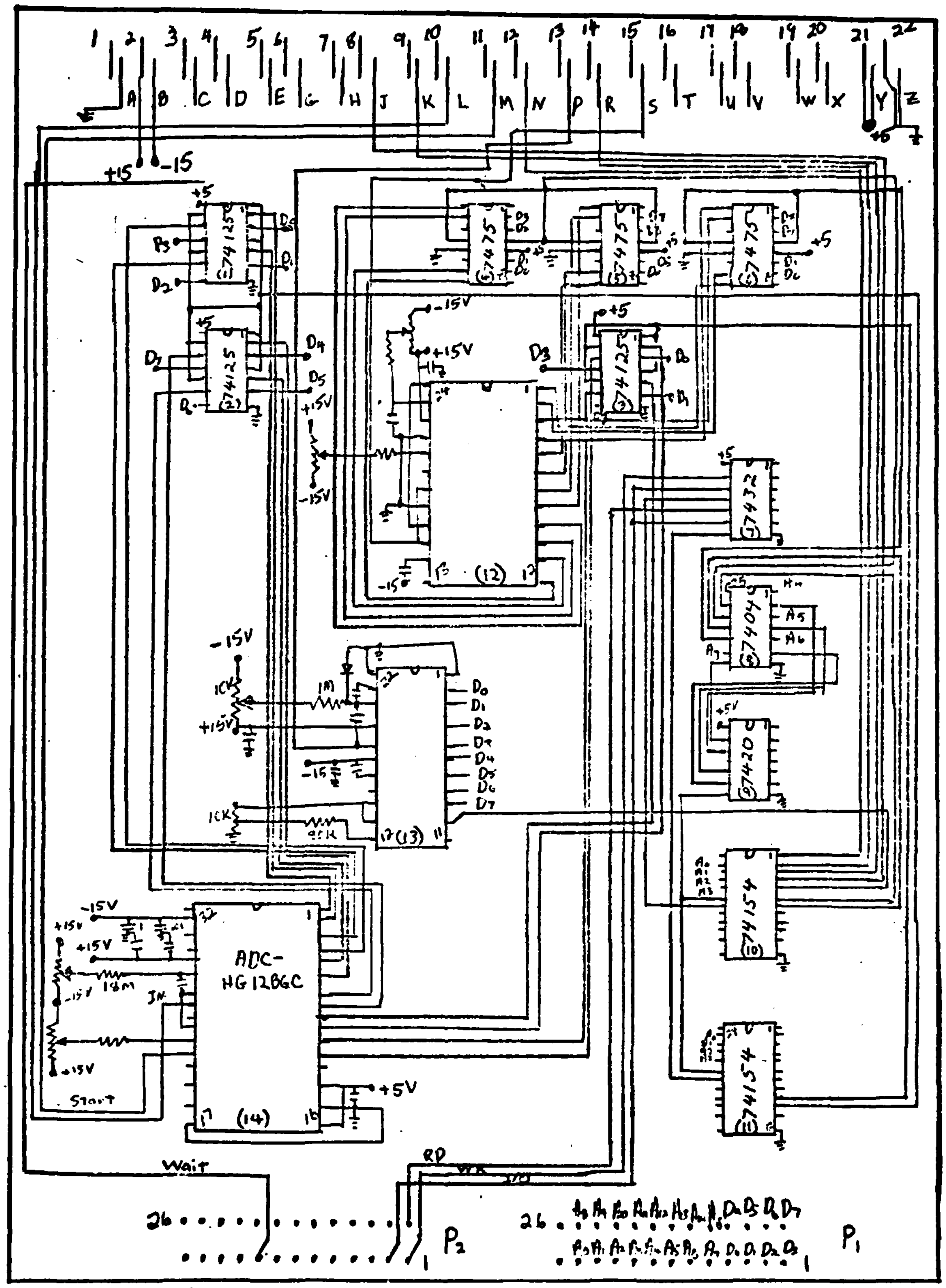


Figure 18. Wire Wrap Board 1 Layout

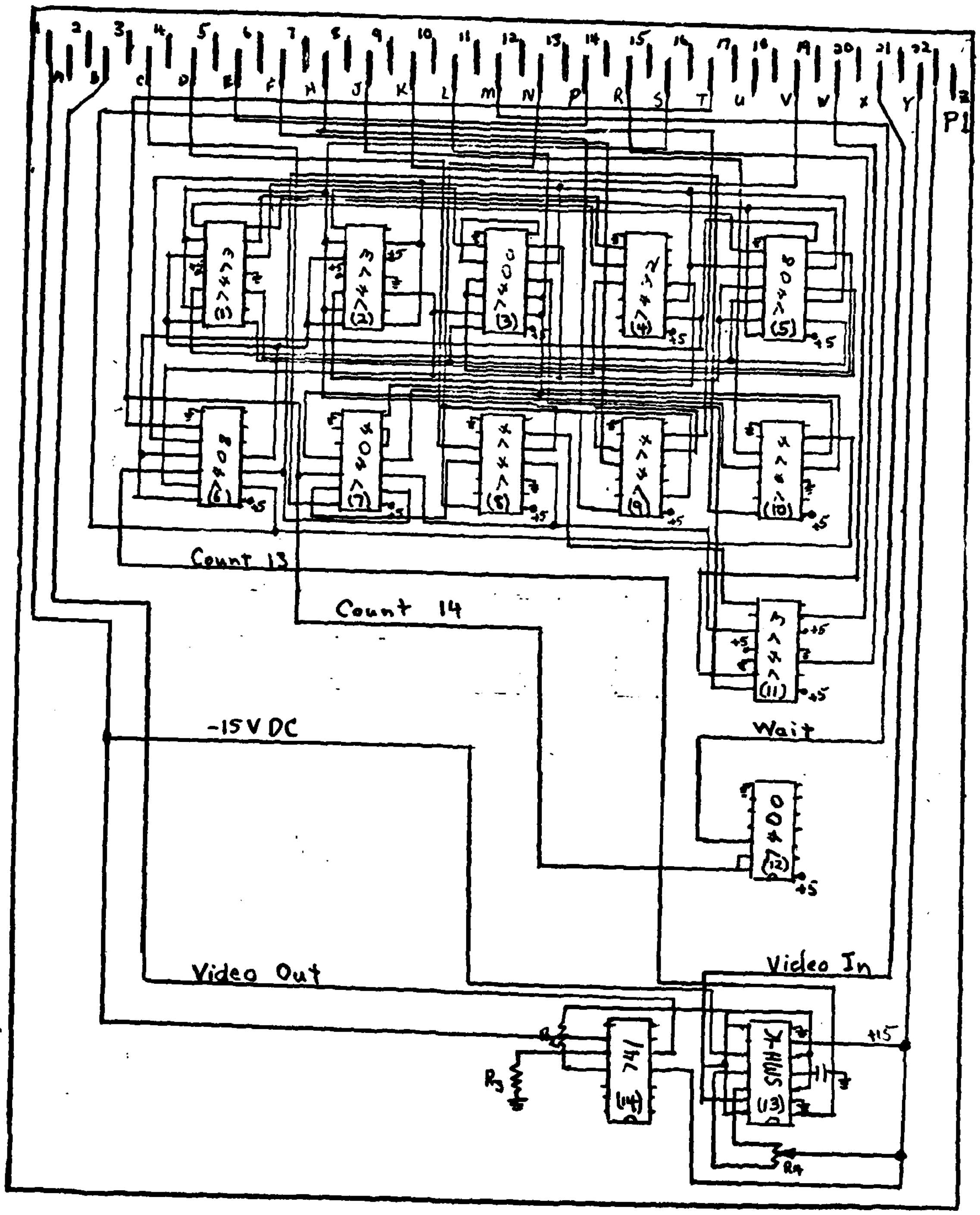


Figure 19. Wire Wrap Board 2 Layout

Vita

Captain Albert L. Lawson was born 16 October 1944 in McKinney, Texas. He graduated from South Oak Cliff High School, Dallas, Texas in 1963. He enlisted in the Air Force in Janurary 1965, and was accepted for the Airman Education and Commissioning Program in August 1972. He attended college at the University of Wyoming from September 1972 to December 1974, graduating with a Batchelor of Science Degree in Electrical Engineering.

Captain Lawson's first assignment after completion of OTS was to the 4900th Test Group at Kirtland AFB, NM. There he worked in the Aircraft Modification Branch. In April 1976 he was transferred to the 6594th Test Group, Hickam AFB, HI, where he worked as a space systems recovery engineer until being assigned to AFIT in May 1980.

Captain Lawson's permanent home address is:

323 Blossom Ave.
Duncanville, TX
75137

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/81D-33	2. GOVT ACCESSION NO. AD A115 502	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Programmed Control of a Sine-Wave Grating Generator for Visual Research		5. TYPE OF REPORT & PERIOD COVERED MS THESIS
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Albert L. Lawson Captain USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project 7071-00-12
11. CONTROLLING OFFICE NAME AND ADDRESS Aerospace Medical Research Laboratory/HEA Wright-Patterson AFB, Ohio 45433		12. REPORT DATE Dec 1981
		13. NUMBER OF PAGES 94
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

15 APR 1982

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

[illegible]

~~Frederick C. Lynch, Major, USAF~~

~~Director of Public~~ Affairs

Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Sine-Wave Grating, Visual Research, Programmed Control of Sine-Wave Gratings

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the development of an automatic controller for an Optronix Optical Signal Generator. The signal generator is used by the Aerospace Medical Research Laboratory/HEA at Wright-Patterson AFB to test Human Visual response to sine-wave gratings. The computer used in development of the controller was a Z80 based Mostek MDX CPU-2. Sensing of the optical signal to be controlled was accomplished by a Reticon RC-100 Series

Circuit Board with a G Series Array Board containing 1024 diode elements. The Reticon Array Board produces an analog video signal for each diode in the array. The computer then selects every fifteenth diode value and stores it, calculates the actual screen brightness and contrast from the stored values, and outputs a correction signal to the generator. The circuits developed include the flags required to synchronize the computer to the Reticon Card, the circuits required to input the desired values for contrast and brightness, and the circuit used to select every fifteenth diode of the array scan to be read. This controller was never fully operational, but all of the major circuits have been tested and do function.

ILMED

8